

7.5

IBM WebSphere MQ 开发应用程序参考

IBM

注

在使用本资料及其支持的产品之前，请阅读第 1331 页的『声明』中的信息。

此版本适用于 IBM® WebSphere MQ V 7 发行版 5 以及所有后续发行版和修订版，直到在新版本中另有声明为止。

当您向 IBM 发送信息时，授予 IBM 以它认为适当的任何方式使用或分发信息的非独占权利，而无需对您承担任何责任。

© Copyright International Business Machines Corporation 2007, 2024.

内容

| | |
|--|----------|
| 开发应用程序参考 | 7 |
| MQI 应用程序参考..... | 7 |
| 代码示例..... | 7 |
| 常量..... | 49 |
| MQI 中使用的数据类型..... | 214 |
| 函数调用..... | 546 |
| 对象的属性..... | 701 |
| 返回码..... | 764 |
| 验证 MQI 选项的规则..... | 765 |
| 发布/预订命令消息..... | 768 |
| 机器编码..... | 785 |
| 报告选项和消息标志..... | 788 |
| 数据转换..... | 791 |
| 指定为 MQRFH2 元素的属性..... | 811 |
| 代码页转换..... | 818 |
| 64 位平台上的编码标准..... | 846 |
| SOAP 引用..... | 850 |
| amqSOAPNETListener: IBM Websphere MQ SOAP Listener for .NET Framework 1 或 2..... | 850 |
| amqswsdl: 为 .NET 服务生成 WSDL..... | 852 |
| amqwclientconfig: 创建客户机部署描述符..... | 853 |
| amqwdeployWMQService: 部署 Web Service 实用程序..... | 853 |
| amqwRegisterdotNet: 注册 IBM WebSphere MQ Transport for SOAP to .NET..... | 861 |
| Apache 软件许可证..... | 861 |
| MQMD SOAP 设置..... | 865 |
| MQRFH2 SOAP 设置..... | 870 |
| runivt: 安装验证测试..... | 871 |
| 保护 IBM WebSphere MQ Web Service..... | 874 |
| SimpleJava 侦听器: IBM Websphere MQ SOAP Axis 侦听器 1.4..... | 877 |
| SOAP 侦听器..... | 879 |
| SOAP 发送方..... | 884 |
| 事务..... | 885 |
| URI 参数..... | 885 |
| W3C SOAP over JMS URI..... | 891 |
| IBM WebSphere MQ Transport for SOAP Web Service..... | 896 |
| IBM WebSphere MQ Transport for SOAP Web Service 客户机..... | 899 |
| 用户出口、API 出口和可安装服务参考..... | 901 |
| MQIEP 接口入口点结构..... | 902 |
| 数据转换出口引用..... | 905 |
| MQ_PUBLISH_EXIT - 发布出口..... | 908 |
| 通道出口调用和数据结构..... | 915 |
| API 出口参考..... | 972 |
| 可安装服务接口参考信息..... | 1028 |
| IBM WebSphere MQ Bridge for HTTP 参考..... | 1087 |
| HTTP DELETE..... | 1087 |
| HTTP GET..... | 1090 |
| HTTP POST..... | 1093 |
| HTTP 头..... | 1096 |
| HTTP 返回码..... | 1111 |
| 支持的消息类型..... | 1118 |
| URI 格式..... | 1119 |
| IBM WebSphere MQ .NET 类和接口..... | 1120 |
| MQAsyncStatus..... | 1120 |

| | |
|---------------------------------------|------|
| MQAuthenticationInformation 记录..... | 1121 |
| MQ 目标..... | 1122 |
| MQEnvironment..... | 1124 |
| MQException..... | 1126 |
| MQGetMessageOptions..... | 1127 |
| MQManagedObject..... | 1130 |
| MQMessage..... | 1132 |
| MQProcess..... | 1142 |
| MQPropertyDescriptor..... | 1145 |
| MQPutMessageOptions..... | 1146 |
| MQQueue..... | 1148 |
| MQQueueManager..... | 1155 |
| MQ 预订..... | 1167 |
| MQTopic..... | 1168 |
| IMQObjectTrigger..... | 1173 |
| MQC..... | 1173 |
| .NET 应用程序的字符集标识..... | 1174 |
| IBM WebSphere MQ C++ 类..... | 1176 |
| MQI 交叉引用..... | 1178 |
| ImqAuthentication 记录..... | 1191 |
| ImqBinary..... | 1193 |
| ImqCache..... | 1195 |
| ImqChannel..... | 1197 |
| ImqCICSBridge 头..... | 1202 |
| ImqDeadLetterHeader..... | 1208 |
| ImqDistribution 列表..... | 1211 |
| ImqError..... | 1212 |
| ImqGetMessageOptions..... | 1213 |
| ImqHeader..... | 1216 |
| ImqIMSBridge 头..... | 1218 |
| ImqItem..... | 1220 |
| ImqMessage..... | 1222 |
| ImqMessage 跟踪程序..... | 1228 |
| ImqNamelist..... | 1231 |
| ImqObject..... | 1233 |
| ImqProcess..... | 1238 |
| ImqPutMessageOptions..... | 1239 |
| ImqQueue..... | 1241 |
| ImqQueue 管理器..... | 1251 |
| ImqReference 头..... | 1266 |
| ImqString..... | 1269 |
| ImqTrigger..... | 1273 |
| ImqWork 头..... | 1276 |
| 针对 Java 库的 IBM WebSphere MQ 类..... | 1277 |
| 用于 JMS 对象的 IBM WebSphere MQ 类的属性..... | 1278 |
| APPLICATIONNAME..... | 1282 |
| ASYNCEXCEPTION..... | 1282 |
| BROKERCCDURSUBQ..... | 1283 |
| BROKERCCSUBQ..... | 1284 |
| BROKERCONQ..... | 1284 |
| BROKERDURSUBQ..... | 1285 |
| BROKERPUBQ..... | 1285 |
| BROKERPUBQMGR..... | 1285 |
| BROKERQMGR..... | 1286 |
| BROKERSUBQ..... | 1286 |
| BROKERVER..... | 1287 |
| CCDTURL..... | 1287 |
| CCSID..... | 1288 |

| | |
|-----------------------------|------|
| 通道..... | 1288 |
| CLEANUP..... | 1289 |
| CLEANUPINT..... | 1289 |
| connectionNameList..... | 1290 |
| CLIENTRECONNECTOPTIONS..... | 1290 |
| CLIENTRECONNECTTIMEOUT..... | 1291 |
| CLIENTID..... | 1291 |
| CLONESUPP..... | 1292 |
| COMPHDR..... | 1292 |
| COMPMSG..... | 1293 |
| CONNOPT..... | 1293 |
| CONNTAG..... | 1294 |
| DESCRIPTION..... | 1294 |
| DIRECTAUTH..... | 1295 |
| ENCODING..... | 1295 |
| EXPIRY..... | 1296 |
| FAILIFQUIESCE..... | 1297 |
| HOSTNAME..... | 1297 |
| LOCALADDRESS..... | 1298 |
| MAPNAMESTYLE..... | 1298 |
| MAXBUFFSIZE..... | 1299 |
| MDREAD..... | 1299 |
| MDWRITE..... | 1300 |
| MDMSGCTX..... | 1300 |
| MSGBATCHSZ..... | 1301 |
| MSGBODY..... | 1301 |
| MSGRETENTION..... | 1302 |
| MSGSELECTION..... | 1302 |
| MULTICAST..... | 1303 |
| OPTIMISTICPUBLICATION..... | 1303 |
| OUTCOMENOTIFICATION..... | 1304 |
| PERSISTENCE..... | 1304 |
| POLLINGINT..... | 1305 |
| 端口..... | 1306 |
| PRIORITY..... | 1306 |
| PROCESSDURATION..... | 1307 |
| PROVIDERVERSION..... | 1307 |
| PROXYHOSTNAME..... | 1308 |
| PROXYPORT..... | 1309 |
| PUBACKINT..... | 1309 |
| PUTASYNCALLOWED..... | 1309 |
| QMANAGER..... | 1310 |
| 队列..... | 1310 |
| READAHEADALLOWED..... | 1311 |
| READAHEADCLOSEPOLICY..... | 1311 |
| RECEIVECCSID..... | 1312 |
| RECEIVECONVERSION..... | 1312 |
| RECEIVEISOLATION..... | 1313 |
| RECEXIT..... | 1313 |
| RECEXITINIT..... | 1314 |
| REPLYTOSTYLE..... | 1314 |
| RESCANINT..... | 1315 |
| SECEXIT..... | 1315 |
| SECEXITINIT..... | 1316 |
| SENDCHECKCOUNT..... | 1316 |
| SENDEXIT..... | 1317 |
| SENDEXITINIT..... | 1317 |
| SHARECONVALLOWED..... | 1318 |

| | |
|-------------------------|-------------|
| SPARSESUBS..... | 1318 |
| SSLCIPHERSUITE..... | 1319 |
| SSLCRL..... | 1319 |
| SSLFIPSREQUIRED..... | 1319 |
| SSLPEERNAME..... | 1320 |
| SSLRESETCOUNT..... | 1320 |
| STATREFRESHINT..... | 1321 |
| SUBSTORE..... | 1321 |
| SYNCPOINTALLGETS..... | 1322 |
| TARGCLIENT..... | 1322 |
| TARGCLIENTMATCHING..... | 1323 |
| TEMPMODEL..... | 1323 |
| TEMPQOPREFIX..... | 1324 |
| TEMPTOPICPREFIX..... | 1324 |
| TOPIC..... | 1324 |
| TRANSPORT..... | 1325 |
| WILDCARDFORMAT..... | 1325 |
| 属性依赖关系..... | 1326 |
| ENCODING 属性..... | 1327 |
| SSL 属性..... | 1328 |
| 声明..... | 1331 |
| 编程接口信息..... | 1332 |
| 商标..... | 1332 |

开发应用程序参考

使用此部分中的信息来帮助您开发 IBM WebSphere MQ 应用程序:

相关任务

[开发应用程序](#)

MQI 应用程序参考

使用本节中提供的链接来帮助您开发 MQI 应用程序:

- [第 7 页的『代码示例』](#)
- [第 49 页的『常量』](#)
- [第 214 页的『MQI 中使用的数据类型』](#)
- [第 546 页的『函数调用』](#)
- [第 701 页的『对象的属性』](#)
- [第 764 页的『返回码』](#)
- [第 765 页的『验证 MQI 选项的规则』](#)
- [第 785 页的『机器编码』](#)
- [第 788 页的『报告选项和消息标志』](#)
- [第 791 页的『数据转换出口』](#)
- [第 811 页的『指定为 MQRFH2 元素的属性』](#)
- [第 818 页的『代码页转换』](#)

相关概念

[第 901 页的『用户出口、API 出口和可安装服务参考』](#)

使用本节中提供的链接来帮助您开发用户出口， API 出口和可安装服务应用程序:

[第 1277 页的『Java 库的 IBM WebSphere MQ 类』](#)

Java 库的 IBM WebSphere MQ 类的位置因平台而异。您启动应用程序时需指定此位置。

相关任务

[开发应用程序](#)

相关参考

[第 850 页的『SOAP 引用』](#)

按字母顺序排列的 WebSphere MQ Transport for SOAP 参考信息。

[第 1087 页的『IBM WebSphere MQ Bridge for HTTP 的参考材料』](#)

IBM WebSphere MQ Bridge for HTTP 的参考主题，按字母顺序排列

[第 1120 页的『IBM WebSphere MQ .NET 类和接口』](#)

IBM WebSphere MQ .NET 类和接口按字母顺序列出。描述了属性，方法和构造函数。

[第 1176 页的『IBM WebSphere MQ C++ 类』](#)

IBM WebSphere MQ C++ 类封装了 IBM WebSphere MQ 消息队列接口 (MQI)。有一个单独的 C++ 头文件 **imqi.hpp**，它涵盖所有这些类。

相关信息

[../com.ibm.mq.javadoc.doc/WMQJMSClasses/index.html](#)

代码示例

使用本部分中的参考信息可以完成满足您业务需求的任务。

C 语言示例

此主题集合主要来自 WebSphere MQ for z/OS 样本应用程序。它们适用于所有平台，但注明的除外。

连接到队列管理器

此示例演示如何使用 MQCONN 调用将程序连接到 z/OS 批处理中的队列管理器。

此抽取来自 WebSphere MQ for z/OS 随附的“浏览”样本应用程序(程序 CSQ4BCA1)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```
#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;
    }
:
}
```

断开与队列管理器的连接

此示例演示如何使用 MQDISC 调用将程序与 z/OS 批处理中的队列管理器断开连接。

此代码抽取中使用的变量是在第 8 页的『连接到队列管理器』中设置的变量。此抽取来自 WebSphere MQ for z/OS 随附的“浏览”样本应用程序(程序 CSQ4BCA1)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```
:
/*                                     */
/*     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;
}
:

```

创建动态队列

此示例演示如何使用 MQOPEN 调用来创建动态队列。

此抽取取自随 WebSphere MQ for z/OS 提供的 Mail Manager 样本应用程序 (程序 CSQ4TCD1)。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
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};
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 */
/*-----*/
MQMErrorHandler( "OPEN TEMPQ", CompCode,
                Reason );
        ErrorFound = TRUE;
}
return ErrorFound;
}
:

```

打开现有队列

此示例演示如何使用 MQOPEN 调用来打开已定义的队列。

此抽取来自 WebSphere MQ for z/OS 随附的 "浏览" 样本应用程序 (程序 CSQ4BCA1)。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

#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 */

```

关闭队列

此示例演示如何使用 MQCLOSE 调用来关闭队列。

此抽取来自 WebSphere MQ for z/OS 随附的 "浏览" 样本应用程序 (程序 CSQ4BCA1)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
/*
/* 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;
}
:

```

使用 MQPUT 放置消息

此示例演示如何使用 MQPUT 调用将消息放入队列。

此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。有关样本应用程序的名称和位置，请参阅 [样本程序 \(平台除外 z/OS\)](#)。

```

:
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 */
    }
}

```

使用 MQPUT1 放置消息

此示例演示如何使用 MQPUT1 调用来打开队列，将单个消息放入队列，然后关闭队列。

此抽取摘自随 WebSphere MQ for z/OS 提供的 Credit Check 样本应用程序 (程序 CSQ4CCB5)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
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);
    }
}

```



```

    }
    else
    {
        sprintf(pBuff, MESSAGE_4_E,
            ERROR_IN_MQGET, CompCode, Reason);
        PrintLine(pBuff);
        RetCode = CSQ4_ERROR;
    }
    :
} /* end of main */

```

使用 *wait* 选项获取消息

此示例演示如何使用 MQGET 调用的 *wait* 选项。

此代码接受截断的消息。此抽取摘自随 WebSphere MQ for z/OS 提供的 Credit Check 样本应用程序 (程序 CSQ4CCB5)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
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();
    }
:

```

使用信令获取消息

信令仅可用于 *WebSphere MQ for z/OS*。

此示例演示如何使用 MQGET 调用来设置信号，以便在合适的消息到达队列时通知您。此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

:
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;
}

```

查询对象的属性

此示例演示如何使用 MQINQ 调用来查询队列的属性。

此抽取来自随 WebSphere MQ for z/OS 提供的 "队列属性" 样本应用程序 (程序 CSQ4CCC1)。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

#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 SelectorTable[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 SelectorTable to the attributes whose      */
/*      status is      */
/*      required      */
/*      - All other variables are already set      */
/*      */
SelectorTable[0] = MQIA_INHIBIT_GET;
SelectorTable[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,
      SelectorTable,
      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 */
}

```

设置队列的属性

此示例演示如何使用 MQSET 调用来更改队列的属性。

此抽取来自随 WebSphere MQ for z/OS 提供的 "队列属性" 样本应用程序 (程序 CSQ4CCC1)。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```
#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 */

```

使用 MQSTAT 检索状态信息

此示例演示如何发出异步 MQPUT 并使用 MQSTAT 检索状态信息。

此抽取取自调用 MQSTAT 样本应用程序 (程序 amqsapt0) 随 WebSphere MQ for Windows 系统提供。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

/*****
/*
/* 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)

```



```

        &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 示例

此主题集合来自 WebSphere MQ for z/OS 样本应用程序。它们适用于所有平台，但注明的除外。

连接到队列管理器

此示例演示如何使用 MQCONN 调用将程序连接到 z/OS 批处理中的队列管理器。

此抽取从 WebSphere MQ for z/OS 随附的“浏览”样本应用程序(程序 CSQ4BVA1)中获取。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

* -----*
* 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.
:

```

断开与队列管理器的连接

此示例演示如何使用 MQDISC 调用将程序与 z/OS 批处理中的队列管理器断开连接。

此代码抽取中使用的变量是在第 23 页的『[连接到队列管理器](#)』中设置的变量。此抽取从 WebSphere MQ for z/OS 随附的“浏览”样本应用程序(程序 CSQ4BVA1)中获取。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
*
* 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.
:

```

创建动态队列

此示例演示如何使用 MQOPEN 调用来创建动态队列。

此抽取摘自随 WebSphere MQ for z/OS 提供的 Credit Check 样本应用程序(程序 CSQ4CVB1)。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
* -----*
* 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 = MQ00-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
*

```

打开现有队列

此示例演示如何使用 MQOPEN 调用来打开现有队列。

此抽取从 WebSphere MQ for z/OS 随附的 "浏览" 样本应用程序 (程序 CSQ4BVA1) 中获取。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
* -----*
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 = MQ00-INQUIRE + MQ00-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

```

关闭队列

此示例演示如何使用 MQCLOSE 调用。

此代码抽取中使用的变量是在第 23 页的『[连接到队列管理器](#)』中设置的变量。此抽取从 WebSphere MQ for z/OS 随附的“浏览”样本应用程序 (程序 CSQ4BVA1) 中获取。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
*
*   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.
*

```

使用 MQPUT 放置消息

此示例演示如何使用上下文的 MQPUT 调用。

此抽取摘自随 WebSphere MQ for z/OS 提供的 Credit Check 样本应用程序 (程序 CSQ4CVB1)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
* -----*
*   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.

```

使用 MQPUT1 放置消息

此示例演示如何使用 MQPUT1 调用。

此抽取来自 WebSphere MQ for z/OS 随附的 Credit Check 样本应用程序 (程序 CSQ4CVB5)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
* -----*
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.
*

```

获取消息

此示例演示如何使用 MQGET 调用从队列中除去消息。

此抽取摘自随 WebSphere MQ for z/OS 提供的 Credit Check 样本应用程序 (程序 CSQ4CVB1)。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
* -----*
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 CMQGMV.

*
* 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.

```

使用 *wait* 选项获取消息

此示例演示如何将 MQGET 调用与 wait 选项配合使用并接受截断的消息。

此抽取来自 WebSphere MQ for z/OS 随附的 Credit Check 样本应用程序 (程序 CSQ4CVB5)。有关其他平台上样本应用程序的名称和位置, 请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```
:
* -----*
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 CMQGMV.
*
*   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 MQ0D-OBJECTNAME TO M02-OBJECTNAME
* PERFORM RECORD-CALL-ERROR
* END-IF.
*
*
*

```

使用信令获取消息

此示例演示如何将 MQGET 调用与信令配合使用。此抽取摘自随 WebSphere MQ for z/OS 提供的 Credit Check 样本应用程序 (程序 CSQ4CVB2)。

信令仅可用于 *WebSphere MQ for z/OS*。

```

*
* -----*
* WORKING-STORAGE SECTION.
* -----*
*
* W00 - General work fields
*
* 01 W00-WAIT-INTERVAL PIC S9(09) BINARY VALUE 30000.
*
* W03 - MQM API fields
*
* 01 W03-HCONN PIC S9(9) BINARY VALUE ZERO.
* 01 W03-HOBJ-REPLYQ PIC S9(9) BINARY.
* 01 W03-COMPCODE PIC S9(9) BINARY.
* 01 W03-REASON PIC S9(9) BINARY.
* 01 W03-DATALEN PIC S9(9) BINARY.
* 01 W03-BUFFLEN PIC S9(9) BINARY.
*
* 01 W03-GET-BUFFER.
* 05 W03-CSQ4BQRM.
* COPY CSQ4VB4.
*
* 05 W03-CSQ4BIIM REDEFINES W03-CSQ4BQRM.
* COPY CSQ4VB1.
*
* 05 W03-CSQ4BPGM REDEFINES W03-CSQ4BIIM.
* COPY CSQ4VB5.
*
*
* API control blocks
*
* 01 MQM-MESSAGE-DESCRIPTOR.
* COPY CMQMDV.
* 01 MQM-GET-MESSAGE-OPTIONS.
* COPY CMQGMV.
*
*
* 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 MQ0D-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
:

```

查询对象的属性

此示例演示如何使用 MQINQ 调用来查询队列的属性。

此抽取从随 WebSphere MQ for z/OS 提供的 "队列属性" 样本应用程序 (程序 CSQ4CVC1) 中获取。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序 \(除 z/OS 以外的平台\)](#)。

```

:
* -----*
*   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.
*
*

```

设置队列的属性

此示例演示如何使用 MQSET 调用来更改队列的属性。

此抽取从随 WebSphere MQ for z/OS 提供的 "队列属性" 样本应用程序 (程序 CSQ4CVC1) 中获取。有关其他平台上样本应用程序的名称和位置，请参阅 [样本程序](#) (除 z/OS 以外的平台)

```

*
* -----*
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 汇编语言示例

此主题集合主要来自 WebSphere MQ for z/OS 样本应用程序。

连接到队列管理器

此示例演示如何使用 MQCONN 调用将程序连接到 z/OS 批处理中的队列管理器。

此抽取来自 WebSphere MQ for z/OS 随附的 "浏览" 样本程序 (CSQ4BAA1)。

```

:
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
*

```

断开与队列管理器的连接

此示例演示如何使用 MQDISC 调用将程序与 z/OS 批处理中的队列管理器断开连接。

此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

:
*
*      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    ,(0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
HCONN   DS      F
COMPCODE DS      F
REASON  DS      F
*
*
LEG3    EQU     *-WKEG3
      END

```

创建动态队列

此示例演示如何使用 MQOPEN 调用来创建动态队列。

此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

:
*
*      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(MQ00_OUTPUT) OPEN FOR OUTPUT AND
*      A     R5,=AL4(MQ00_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,CALLST)
*
      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
*
*   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 ,(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L LIST FORM
                                                OF CALL
                                                MACRO
*
:
      END

```

打开现有队列

此示例演示如何使用 MQOPEN 调用来打开已定义的队列。

它显示了如何指定两个选项。此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

:
*
*   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,CALLST)
*
        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
*
*   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
*
CALLST  CALL  ,(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L  LIST FORM
                                                OF CALL
                                                MACRO
*
*
*
        END

```

关闭队列

此示例演示如何使用 MQCLOSE 调用来关闭队列。

此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

*
*
*   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,CALLST)
*
        LA    R5,MQCC_OK
        C     R5,COMPCODE
        BNE  BADCALL
*
*
BADCALL DS    0H
*
*
*   CONSTANTS
*
        CMQA
*
*   WORKING STORAGE (REENTRANT)

```



```

*
WEG4      DSECT
*
CALLLST   CALL  ,(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
HCONN     DS    F
HOBJ      DS    F
OPTIONS   DS    F
COMPCODE  DS    F
REASON    DS    F
*
*
LEG4      EQU   *-WKEG4
          END

```

使用 MQPUT 放置消息

此示例演示如何使用 MQPUT 调用将消息放入队列。

此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

:
*      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
:

```

*


```

                LMQPMO,                X
                BUFFERLENGTH,          X
                BUFFER,                 X
                COMPCODE,               X
                REASON),VL,MF=(E,CALLST)
*
        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),VL,MF=L
*
...
        END

```

获取消息

此示例演示如何使用 MQGET 调用从队列中除去消息。

此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

...
*
*      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 , (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
:
END

```

使用 *wait* 选项获取消息

此示例演示如何使用 MQGET 调用的 *wait* 选项。

此代码接受截断的消息。此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

:
*   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,CALLST)
*
LA   R5,MQCC_OK              DID THE MQGET REQUEST
C    R5,COMPCODE             WORK OK?
BE   GETOK                   YES, SO GO AND PROCESS.
LA   R5,MQCC_WARNING        NO, SO CHECK FOR A WARNING.
C    R5,COMPCODE             IS THIS A WARNING?
BE   CHECK_W                 YES, SO CHECK THE REASON.
*
LA   R5,MQRC_NO_MSG_AVAILABLE  IT MUST BE AN ERROR.
                                        IS IT DUE TO AN EMPTY
C    R5,REASON               QUEUE?
BE   NOMSG                   YES, SO HANDLE THE ERROR
B    BADCALL                 NO, SO GO TO ERROR ROUTINE
*
CHECK_W DS  0H
      LA  R5,MQRC_TRUNCATED_MSG_ACCEPTED  IS THIS A
                                        TRUNCATED
C      R5,REASON                          MESSAGE?
BE   GETOK                                YES, SO GO AND PROCESS.
B    BADCALL                              NO, SOME OTHER WARNING
*
NOMSG DS  0H
:
GETOK  DS  0H
:

```

```

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

```

```

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

```

使用信令获取消息

此示例演示如何使用 MQGET 调用来设置信号，以便在合适的消息到达队列时通知您。

此抽取并非从 WebSphere MQ 随附的样本应用程序中获取。

```

:
*
* 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
       WGM0,                  X
       BUFFLEN,               X
       BUFFER,                X
       DATALEN,              X
       COMPCODE,              X
       REASON),               X
      VL,MF=(E,CALLST)

*
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
*

```



```

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?
:

```

常量

使用本部分中的参考信息可以完成满足您业务需求的任务。

IBM WebSphere MQ COPY ， 头， 包含和模块文件

此信息是通用编程接口信息。

本部分包含帮助您将 MQI 用于各种编程语言的信息， 如下所示。

C 头文件

提供了头文件以帮助您编写使用 MQI 的 C 应用程序。 下表汇总了这些头文件:

表 1: C 头文件-调用原型, 数据类型, 返回码, 常量和结构

| 文件名 | 描述 | IBM i | UNIX 和 Linux® 系统 | Windows | z/OS |
|-------------------------------|-----------------------------------|-------|------------------|---------|------|
| 调用原型, 数据类型, 返回码, 常量和结构 | | | | | |
| CMQC | MQI 定义 | C | C | C | C |
| CMQBC | MQAI 定义 | C | C | C | |
| CMQEC | 接口入口点定义 (包括 CMQC , CMQXC 和 CMQZC) | | C | C | |
| CMQCFC | PCF 定义 | C | C | C | C |
| CMQPSC | 发布/预订定义 | C | C | C | C |
| CMQXC | 通道和出口定义 | C | C | C | C |
| CMQZC | 可安装服务定义 | C | C | C | |
| 键: C= 提供的文件 | | | | | |

COBOL 副本文件

提供了各种 COPY 文件来帮助您编写使用 MQI 的 COBOL 应用程序。下表汇总了这些文件:

表 2: COBOL 副本文件-返回码, 常量和结构

| 文件名 | 描述 | IBM i | UNIX 系统 | Windows | z/OS |
|---------------|-----------------------|-------|---------|---------|------|
| 返回码和常量 | | | | | |
| CMQx | MQI 定义 | V | V | V | V |
| CMQCFx | PCF 定义 | V | V | V | V |
| CMQPSx | 发布/预订定义 | V | V | V | V |
| CMQXx | 通道和出口定义 | V | V | V | V |
| 结构 | | | | | |
| CMQAIRx | MQAIR-认证信息记录 | | V L | V L | |
| CMQBOx | MQBO-开始选项 | V L | V L | V L | |
| CMQCDx | MQCD-通道定义 | V L | V L | V L | V L |
| CMQCFBFx | MQCFBF-PCF 字节字符串过滤器参数 | V L | V L | V L | V L |
| CMQCFBSx | MQCFBS-PCF 字节字符串参数 | V L | V L | V L | V L |
| CMQCFGRx | MQCFGR-PCF 组参数 | V L | V L | V L | V L |
| CMQCFHx | MQCFH-PCF 头 | V L | V L | V L | V L |
| CMQCFIFx | MQCFIF-PCF 整数过滤器参数 | V L | V L | V L | V L |
| CMQCFILx | MQCFIL-PCF 整数列表参数 | V L | V L | V L | V L |
| CMQCFINx | MQCFIN-PCF 整数参数 | V L | V L | V L | V L |
| CMQCFSFx | MQCFSF-PCF 字符串过滤器参数 | V L | V L | V L | V L |
| CMQCFSLx | MQCFSL-PCF 字符串列表参数 | V L | V L | V L | V L |

表 2: COBOL 副本文件-返回码, 常量和结构 (继续)

| 文件名 | 描述 | IBM i | UNIX 系统 | Windows | z/OS |
|----------|--------------------------|-------|---------|---------|------|
| CMQCFSTx | MQCFST-PCF 字符串参数 | V L | V L | V L | V L |
| CMQCFXLx | MQCFIL64 -PCF 64 位整数列表参数 | V L | V L | V L | V L |
| CMQCFXNx | MQCFIN64 -PCF 64 位整数参数 | V L | V L | V L | V L |
| CMQCHRVx | MQCHARV-可变长度字符串 | V L | V L | V L | V L |
| CMQCIHx | MQCIH- CICS 网桥头 | V L | V L | V L | V L |
| CMQCN0x | MQCNO - 连接选项 | V L | V L | V L | V L |
| CMQCSPx | MQCSP-安全性参数 | V L | V L | V L | V L |
| CMQCXPx | MQCXP-通道出口参数 | V L | | | V L |
| CMQDHx | MQDH - 分发头 | V L | V L | V L | V L |
| CMQDLHx | MQDLH-死信头 | V L | V L | V L | V L |
| CMQDXPx | MQDXP-数据转换出口参数 | V L | | V L | |
| CMQEPHx | MQEPH-嵌入式 PCF 头 | V L | V L | V L | V L |
| CMQGM0x | MQGMO-获取消息选项 | V L | V L | V L | V L |
| CMQIIHx | MQIIH- IMS 信息头 | V L | V L | V L | V L |
| CMQMDx | MQMD - 消息描述符 | V L | V L | V L | V L |
| CMQMD1x | MQMD1 -消息描述符版本 1 | V L | V L | V L | V L |
| CMQMD2x | MQMD2 -消息描述符版本 2 | V L | V L | V L | V L |
| CMQMDEx | MQMDE-消息描述符已扩展 | V L | V L | V L | V L |
| CMQODx | MQOD-对象描述符 | V L | V L | V L | V L |
| CMQORx | MQOR-对象记录 | V L | V L | V L | V L |
| CMQPM0x | MQPMO-放置消息选项 | V L | V L | V L | V L |
| CMQRFHx | MQRFH - 规则和格式化头 | V L | V L | V L | V L |
| CMQRFH2x | MQRFH2 -规则和格式化头 2 | V L | V L | V L | V L |
| CMQRMHx | MQRMH - 参考消息头 | V L | V L | V L | V L |
| CMQRRx | MQRR-响应记录 | V L | V L | V L | |
| CMQSCOx | MQSCO-SSL 配置选项 | | V L | V L | |
| CMQTMx | MQTM-触发器消息 | V L | | V L | V L |
| CMQTMcx | MQTMC-触发器消息字符 | V L | V L | | |
| CMQTM2x | MQTMC2 -触发器消息 2 字符 | V L | V L | V L | V L |
| CMQWIHx | MQWIH - 工作信息头 | V L | V L | V L | V L |
| CMQXQHx | MQXQH - 传输队列头 | V L | V L | V L | V L |

关键字:

- 提供了初始值的文件, x = V
- 未提供初始值的文件, x = L

PL/I 包含文件

为 PL/I 编程语言提供了以下 INCLUDE 文件。这些文件仅在 z/OS 上可用。

| 表 3: PL/I include 文件-数据类型, 返回码, 常量和结构 | | | | | |
|---------------------------------------|---------|-------|---------|---------|------|
| 文件名 | 描述 | IBM i | UNIX 系统 | Windows | z/OS |
| 数据类型, 返回码, 常量和结构 | | | | | |
| CMQP | MQI 定义 | | | | P |
| CMQCFP | PCF 定义 | | | | P |
| CMQEPP | 入口点定义 | | | | P |
| CMQPSP | 发布/预订定义 | | | | P |
| CMQXP | 通道和出口定义 | | | | P |
| 键: P= 提供的文件 | | | | | |

RPG 副本文件

为 RPG 编程语言提供了以下 COPY 文件。这些文件仅在 IBM i 上可用。

| 表 4: RPG 副本文件-返回码, 常量和结构 | | | | | |
|--------------------------|-----------------------|-------|---------|---------|------|
| 文件名 | 描述 | IBM i | UNIX 系统 | Windows | z/OS |
| 返回码和常量 | | | | | |
| CMQx | MQI 定义 | G R | | | |
| CMQCFx | PCF 定义 | G | | | |
| CMQPSx | 发布/预订定义 | G | | | |
| CMQXx | 通道和出口定义 | G R | | | |
| 结构 | | | | | |
| CMQBOx | MQBO-开始选项 | G H | | | |
| CMQCDx | MQCD-通道定义 | G H R | | | |
| CMQCFBFx | MQCFBF-PCF 字节字符串过滤器参数 | G H | | | |
| CMQCFBSx | MQCFBS-PCF 字节字符串参数 | G H | | | |
| CMQCFGRx | MQCFGR-PCF 组参数 | G H | | | |
| CMQCFHx | MQCFH-PCF 头 | G H | | | |
| CMQCFIFx | MQCFIF-PCF 整数过滤器参数 | G H | | | |
| CMQCFILx | MQCFIL-PCF 整数列表参数 | G H | | | |
| CMQCFINx | MQCFIN-PCF 整数参数 | G H | | | |
| CMQCFSFx | MQCFSF-PCF 字符串过滤器参数 | G H | | | |
| CMQCFSLx | MQCFSL-PCF 字符串列表参数 | G H | | | |
| CMQCFSTx | MQCFST-PCF 字符串参数 | G H | | | |

表 4: RPG 副本文件-返回码, 常量和结构 (继续)

| 文件名 | 描述 | IBM i | UNIX 系统 | Windows | z/OS |
|-----------|--------------------------|-------|---------|---------|------|
| CMQCFXLx | MQCFIL64 -PCF 64 位整数列表参数 | G H | | | |
| CMQCFXNx | MQCFIN64 -PCF 64 位整数参数 | G H | | | |
| CMQCHARVx | MQCHARV-可变长度字符串 | G H | | | |
| CMQCIHx | MQCIH- CICS 网桥头 | G H | | | |
| CMQCNOx | MQCNO - 连接选项 | G H | | | |
| CMQCSPx | MQCSP-安全性参数 | G H | | | |
| CMQCXPx | MQCXP-通道出口参数 | G H R | | | |
| CMQDHx | MQDH - 分发头 | G H R | | | |
| CMQDLHx | MQDLH-死信头 | G H R | | | |
| CMQDXPx | MQDXP-数据转换出口参数 | G H R | | | |
| CMQEPHx | MQEPH-嵌入式 PCF 头 | G H | | | |
| CMQGMox | MQGMO-获取消息选项 | G H R | | | |
| CMQIIHx | MQIIH- IMS 信息头 | G H R | | | |
| CMQMDx | MQMD - 消息描述符 | G H R | | | |
| CMQMD1x | MQMD1 -消息描述符版本 1 | G H R | | | |
| CMQMD2x | MQMD2 -消息描述符版本 2 | G H | | | |
| CMQMDEx | MQMDE-消息描述符已扩展 | G H R | | | |
| CMQODx | MQOD-对象描述符 | G H R | | | |
| CMQORx | MQOR-对象记录 | G H R | | | |
| CMQPMox | MQPMO-放置消息选项 | G H R | | | |
| CMQPXPx | MQPXP-发布/预订路由出口参数 | G H | | | |
| CMQRFHx | MQRFH - 规则和格式化头 | G H | | | |
| CMQRFH2x | MQRFH2 -规则和格式化头 2 | G H | | | |
| CMQRMHx | MQRMH - 参考消息头 | G H R | | | |
| CMQRRx | MQRR-响应记录 | G H R | | | |
| CMQTMx | MQTM-触发器消息 | G H R | | | |
| CMQTMcx | MQTMC-触发器消息字符 | G H R | | | |
| CMQTM2x | MQTMC2 -触发器消息 2 字符 | G H R | | | |
| CMQWIHx | MQWIH - 工作信息头 | G H | | | |
| CMQXQHx | MQXQH - 传输队列头 | G H R | | | |

表 4: RPG 副本文件-返回码, 常量和结构 (继续)

| 文件名 | 描述 | IBM i | UNIX 系统 | Windows | z/OS |
|---|----|-------|---------|---------|------|
| 关键字: | | | | | |
| <ul style="list-style-type: none"> • 用于静态链接的文件, 已初始化, 提供 x = G • 用于静态链接的文件, 未初始化, 提供 x = H • 用于动态链接的文件, 已初始化, 已提供, x = R | | | | | |

Visual Basic 模块文件

提供了头 (或表单) 文件以帮助您编写使用 MQI 的 Visual Basic 应用程序。这些头文件仅以 32 位版本提供, 并在表中进行了概述:

表 5: Visual Basic 模块文件-调用声明, 数据类型, 返回码, 常量和结构

| 文件名 | 描述 | IBM i | UNIX and Linux 系统 | Windows | z/OS |
|------------------------|---------|-------|-------------------|---------|------|
| 调用声明, 数据类型, 返回码, 常量和结构 | | | | | |
| CMQB | MQI 定义 | | | B | |
| CMQBB | MQAI 定义 | | | B | |
| CMQCFB | PCF 定义 | | | B | |
| CMQXB | 通道和出口定义 | | | B | |
| 键: B= 提供的文件 | | | | | |

MQ_* (字符串长度)

表 6: 常量值

| : | 小数制值 | 十六进制值 |
|-------------------------------|------|-------------|
| 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' |

| 表 6: 常量值 (继续) | | |
|-------------------------------|--|-------------|
| : | 小数值 | 十六进制值 |
| MQ_BRIDGE_NAME_LENGTH | 24 | X'00000018' |
| MQ_CANCEL_CODE_LENGTH | 4 | X'00000004' |
| MQ_CF_STRUC_DESC_LENGTH | 64 | X'00000040' |
| MQ_CF_STRUC_NAME_LENGTH | 12 | X'0000000C' |
| MQ_CHANNEL_DATE_LENGTH | 12 | X'0000000C' |
| MQ_CHANNEL_DESC_LENGTH | 64 | X'00000040' |
| MQ_CHANNEL_NAME_LENGTH | 20 | X'00000014' |
| MQ_CHANNEL_TIME_LENGTH | 8 | X'00000008' |
| MQ_CHINIT_SERVICE_PARM_LENGTH | 32 | X'00000020' |
| MQ_CICS_FILE_NAME_LENGTH | 8 | X'00000008' |
| MQ_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_专有名称长度 | 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' |

| 表 6: 常量值 (继续) | | |
|------------------------------|--|-------------|
| : | 小数值 | 十六进制值 |
| 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' |
| 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' |

| 表 6: 常量值 (继续) | | |
|--------------------------------|-------|-------------|
| : | 小数值 | 十六进制值 |
| MQ_SERVICE_NAME_LENGTH | 32 | X'00000020' |
| MQ_SERVICE_PATH_LENGTH | 255 | X'000000FF' |
| MQ_SERVICE_STEP_LENGTH | 8 | X'00000008' |
| MQ_SHORT_CONN_NAME_LENGTH | 20 | X'00000014' |
| MQ_SHORT_DNAME_LENGTH | 256 | X'00000100' |
| MQ_SSL_CIPHER_SPEC_LENGTH | 32 | X'00000020' |
| MQ_SSL_CRYPTO_HARDWARE_LENGTH | 256 | X'00000100' |
| MQ_SSL_HANDSHAKE_STAGE_LENGTH | 32 | X'00000020' |
| MQ_SSL_KEY_LIBRARY_LENGTH | 44 | X'0000002C' |
| MQ_SSL_KEY_MEMBER_LENGTH | 8 | X'00000008' |
| MQ_SSL_KEY_REPOSITORY_LENGTH | 256 | X'00000100' |
| MQ_SSL_PEER_NAME_LENGTH | 1024 | X'00000400' |
| MQ_SSL_SHORT_PEER_NAME_LENGTH | 256 | X'00000100' |
| MQ_START_CODE_LENGTH | 4 | X'00000004' |
| MQ_STORAGE_CLASS_DESC_LENGTH | 64 | X'00000040' |
| MQ_STORAGE_CLASS_LENGTH | 8 | X'00000008' |
| MQ_SUB_IDENTITY_LENGTH | 128 | X'00000080' |
| 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' |

| 表 6: 常量值 (继续) | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQ_XCF_GROUP_NAME_LENGTH | 8 | X'00000008' |
| MQ_XCF_MEMBER_NAME_LENGTH | 16 | X'00000010' |

MQ_* (命令格式字符串长度)

| 表 7: 常量值 | | |
|-----------------------------|-------|-------------|
| : | 小数值 | 十六进制值 |
| 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'00004000' |
| 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_* (API 出口链区域头结构)

| 表 8: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQACH_STRUC_ID | "ACH↵" |
| MQACH_STRUC_ID_ARRAY | 'A', 'C', 'H', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 9: 常量值 | | |
|-----------------------|--|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (记帐令牌)

| 表 10: 常量名称和值 | |
|------------------|--------------------------|
| : | 值 |
| MQACT_NONE | X'00...00' (32 个空值) |
| MQACT_NONE_ARRAY | '\0', '\0', ... (32 个空值) |

MQACT_* (命令格式操作选项)

| 表 11: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQACT_FORCE_REMOVE | 1 | X'00000001' |
| MQACT_ADVANCE_LOG | 2 | X'00000002' |
| MQACT_COLLECT_STATISTICS | 3 | X'00000003' |
| MQACT_PUBSUB | 4 | X'00000004' |

MQACTP_* (操作)

| 表 12: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQACTP_NEW | 0 | X'00000000' |
| MQACTP_FORWARD | 1 | X'00000001' |
| MQACTP_REPLY | 2 | X'00000002' |
| MQACTP_REPORT | 3 | X'00000003' |

MQACTT_* (记帐令牌类型)

| 表 13: 常量值 | |
|---------------------|-------|
| : | 十六进制值 |
| MQACTT_UNKNOWN | X'00' |
| MQACTT_CICS_LUOW_ID | X'01' |

| 表 13: 常量值 (继续) | |
|----------------------------|-------|
| : | 十六进制值 |
| MQACTT_OS2_DEFAULT | X'04' |
| MQACTT_DOS_DEFAULT | X'05' |
| MQACTT_UNIX_NUMERIC_ID | X'06' |
| MQACTT_OS400_ACCOUNT_TOKEN | X'08' |
| MQACTT_WINDOWS_DEFAULT | X'09' |
| MQACTT_NT_SECURITY_ID | X'0B' |
| MQACTT_USER | X'19' |

MQADOPT_* (采用新的 MCA 检查和采用新的 MCA 类型)

采用新的 MCA 检查

| 表 14: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

采用新的 MCA 类型

| 表 15: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (认证信息记录结构)

| 表 16: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQAIR_STRUC_ID | "AIR↵" |
| MQAIR_STRUC_ID_ARRAY | 'A', 'I', 'R', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 17: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQAIR_VERSION_1 | 1 | X'00000001' |
| MQAIR_VERSION_2 | 2 | X'00000002' |
| MQAIR_CURRENT_VERSION | 2 | X'00000002' |

MQAIT_* (认证信息类型)

| 表 18: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQAIT_ALL | 0 | X'00000000' |
| MQAIT_CRL_LDAP | 1 | X'00000001' |
| MQAIT_OCSP | 2 | X'00000002' |

MQAS_* (命令格式异步状态值)

| 表 19: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQAS_NONE | 0 | X'00000000' |
| MQAS_STARTED | 1 | X'00000001' |
| MQAS_START_WAIT | 2 | X'00000002' |
| MQAS_STOPPED | 3 | X'00000003' |
| 已暂挂 MQAS_SUSPENDED | 4 | X'00000004' |
| MQAS_SUSPENDED_TEMPORARY | 5 | X'00000005' |
| MQAS_ACTIVE | 6 | X'00000006' |
| MQAS_INACTIVE | 7 | X'00000007' |

MQAT_* (放置应用程序类型)

| 表 20: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 20: 常量值 (继续) | | |
|------------------------|--|-------------|
| : | 小数值 | 十六进制值 |
| MQAT_VOS | 14 | X'0000000E' |
| MQAT_OPEN_TP1 | 15 | X'0000000F' |
| MQAT_VM | 18 | X'00000012' |
| MQAT_IMS_BRIDGE | 19 | X'00000013' |
| MQAT_XCF | 20 | X'00000014' |
| MQAT_CICS_BRIDGE | 21 | X'00000015' |
| MQAT_NOTES_AGENT | 22 | X'00000016' |
| MQAT_TPF | 23 | X'00000017' |
| 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 | 99999999 | X'3B9AC9FF' |

MQAUTH_* (命令格式权限值)

| 表 21: 常量值 | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 21: 常量值 (继续) | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQAUTH_SET | 14 | X'0000000E' |
| MQAUTH_SET_ALL_CONTEXT | 15 | X'0000000F' |
| MQAUTH_SET_IDENTITY_CONTEXT | 16 | X'00000010' |
| MQAUTH_CONTROL | 17 | X'00000011' |
| MQAUTH_CONTROL_EXTENDED | 18 | X'00000012' |
| MQAUTH_PUBLISH | 19 | X'00000013' |
| MQAUTH_SUBSCRIBE | 20 | X'00000014' |
| MQAUTH_RESUME | 21 | X'00000015' |
| MQAUTH_SYSTEM | 22 | X'00000016' |

MQAUTHOPT_* (命令格式权限选项)

| 表 22: 常量值 | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_通配符 | 64 | X'00000040' |
| MQAUTHOPT_NAME_EXPLICIT | 16 | X'00000010' |

MQAXC_* (API 出口上下文结构)

| 表 23: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQAXC_STRUC_ID | "AXC↵" |
| MQAXC_STRUC_ID_ARRAY | 'A', 'X', 'C', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 24: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQAXC_VERSION_1 | 1 | X'00000001' |
| MQAXC_CURRENT_VERSION | 1 | X'00000001' |

MQAXP_* (API 出口参数结构)

| 表 25: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQAXP_STRUC_ID | "AXP↵" |
| MQAXP_STRUC_ID_ARRAY | 'A', 'X', 'P', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 26: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQAXP_VERSION_1 | 1 | X'00000001' |
| MQAXP_VERSION_2 | 2 | X'00000002' |
| MQAXP_CURRENT_VERSION | 2 | X'00000002' |

MQBA_* (字节属性选择器)

| 表 27: 常量值 | | |
|------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQBA_FIRST | 6001 | X'00001771' |
| MQBA_LAST | 8000 | X'00001F40' |

MQBACF_* (命令格式字节参数类型)

| 表 28: 常量值 | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (mqAdd 字符串和 mqSet 字符串的缓冲区长度)

| 表 29: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQBL_NULL_TERMINATED | -1 | X'FFFFFFFF' |

MQBMHO_* (缓冲区到消息句柄选项和结构)

缓冲区到消息句柄选项结构

| 表 30: 常量结构 | |
|-----------------------|--------------------|
| : | 结构 |
| MQBMHO_STRUC_ID | "BMHO" |
| MQBMHO_STRUC_ID_ARRAY | 'B', 'M', 'H', 'O' |

注: 符号 - 表示单个空白字符。

| 表 31: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQBMHO_VERSION_1 | 1 | X'00000001' |
| MQBMHO_CURRENT_VERSION | 1 | X'00000001' |

缓冲区到消息句柄选项

| 表 32: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQBMHO_NONE | 0 | X'00000000' |
| MQBMHO_DELETE_PROPERTIES | 1 | X'00000001' |

MQBND_* (缺省绑定)

| 表 33: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQBND_BIND_ON_OPEN | 0 | X'00000000' |
| MQBND_BIND_NOT_FIXED | 1 | X'00000001' |
| MQBND_BIND_ON_GROUP | 2 | X'00000002' |

MQBO_* (开始选项和结构)

开始选项结构

| 表 34: 常量结构 | |
|---------------------|--------------------|
| : | 结构 |
| MQBO_STRUC_ID | "BO--" |
| MQBO_STRUC_ID_ARRAY | 'B', 'O', '-', '-' |

注: 符号 - 表示单个空白字符。

| 表 35: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQBO_VERSION_1 | 1 | X'00000001' |
| MQBO_CURRENT_VERSION | 1 | X'00000001' |

开始选项

| 表 36: 常量值 | | |
|-----------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQBO_NONE | 0 | X'00000000' |

MQBT_* (命令格式网桥类型)

| 表 37: 常量值 | | |
|-----------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQBT_OTMA | 1 | X'00000001' |

MQCA_* (字符属性选择器)

| 表 38: 常量值 | | |
|----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 38: 常量值 (继续) | | |
|---------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 38: 常量值 (继续) | | |
|----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式字符参数类型)

| 表 39: 常量值 | | |
|--------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQCACF_FIRST | 3001 | X'00000BB9' |

| 表 39: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 39: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 39: 常量值 (继续) | | |
|------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 39: 常量值 (继续) | | |
|---------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 39: 常量值 (继续) | | |
|---------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式字符通道参数类型)

| 表 40: 常量值 | | |
|----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 40: 常量值 (继续) | | |
|-----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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 信息头 ADS 描述符)

| 表 41: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCADSD_NONE | 0 | X'00000000' |
| MQCADSD_SEND | 1 | X'00000001' |
| MQCADSD_RECV | 16 | X'00000010' |
| MQCADSD_MSGFORMAT | 256 | X'00000100' |

MQCAFTY_* (连接亲缘关系值)

| 表 42: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCAFTY_NONE | 0 | X'00000000' |
| 首选 MQCAFTY_PREFERRED | 1 | X'00000001' |

MQCAMO_* (命令格式字符监视参数类型)

| 表 43: 常量值 | | |
|-------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (MQCBC 常量结构)

| 表 44: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQCBC_STRUC_ID | "CBC~" |
| MQCBC_STRUC_ID_ARRAY | 'C', 'B', 'C', '~' |

注: 符号 ~ 表示单个空白字符。

| 表 45: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCBC_VERSION_1 | 1 | X'00000001' |
| MQCBC_CURRENT_VERSION | 1 | X'00000001' |

MQCBCF_* (MQCBC 常量标志)

| 表 46: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCBCF_NONE | 0 | X'00000000' |
| MQCBCF_READA_BUFFER_EMPTY | 1 | X'00000001' |

MQCBCT_* (MQCBC 常量回调类型)

| 表 47: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCBCT_START_CALL | 1 | X'00000001' |
| MQCBCT_STOP_CALL | 2 | X'00000002' |
| MQCBCT_REGISTER_CALL | 3 | X'00000003' |

| 表 47: 常量值 (继续) | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (MQCBD 常量结构)

| 表 48: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQCBD_STRUC_ID | "CBD↵" |
| MQCBD_STRUC_ID_ARRAY | 'C','B','D','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 49: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCBD_VERSION_1 | 1 | X'00000001' |
| MQCBD_CURRENT_VERSION | 1 | X'00000001' |

MQCBDO_* (MQCBD 常量回调选项)

| 表 50: 常量值 | | |
|--------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (mqCreateBag 的 Create-Bag 选项)

| 表 51: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_禁止 | 0 | X'00000000' |
| MQCBO_REORDER_AS_REQUIRED | 4 | X'00000004' |
| MQCBO_DO_NOT_REORDER | 0 | X'00000000' |

| 表 51: 常量值 (继续) | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCBO_CHECK_SELECTORS | 8 | X'00000008' |
| MQCBO_DO_NOT_CHECK_SELECTORS | 0 | X'00000000' |

MQCBT_* (MQCBD 常量这是回调函数的类型)

| 表 52: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCBT_MESSAGE_CONSUMER | 1 | X'00000001' |
| MQCBT_EVENT_HANDLER | 2 | X'00000002' |

MQCC_* (完成代码)

| 表 53: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCC_OK | 0 | X'00000000' |
| MQCC_WARNING | 1 | X'00000001' |
| MQCC_FAILED | 2 | X'00000002' |
| MQCC_UNKNOWN | -1 | X'FFFFFFFF' |

MQCCSI_* (编码字符集标识)

| 表 54: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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'FFFFFFF' |
| MQCCSI_APPL | -3 | X'FFFFFFFD' |

MQCCT_* (CICS 信息头会话式任务选项)

| 表 55: 常量值 | | |
|-----------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCCT_YES | 1 | X'00000001' |
| MQCCT_NO | 0 | X'00000000' |

MQCD_* (通道定义结构)

| 表 56: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCD_VERSION_1 | 1 | X'00000001' |
| MQCD_VERSION_2 | 2 | X'00000002' |
| MQCD_VERSION_3 | 3 | X'00000003' |
| MQCD_VERSION_4 | 4 | X'00000004' |

| 表 56: 常量值 (继续) | | |
|----------------------|--|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (通道数据转换)

| 表 57: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCDC_SENDER_CONVERSION | 1 | X'00000001' |
| MQCDC_NO_SENDER_CONVERSION | 0 | X'00000000' |

MQCERT_* (证书验证策略类型)

| | | |
|----------------------------|---|-------------|
| 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_* (功能标志)

| 表 58: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCF_NONE | 0 | X'00000000' |
| MQCF_DIST_LISTS | 1 | X'00000001' |

MQCFAC_* (CICS 信息头工具)

| 表 59: 常量名称和值 | |
|--------------|--------------------|
| : | 十六进制值 |
| MQCFAC_NONE | X'00...00' (8 个空值) |

| 表 59: 常量名称和值 (继续) | |
|-------------------|-------------------------|
| : | 十六进制值 |
| MQCFAC_NONE_ARRAY | '\0', '\0', ... (8 个空值) |

MQCFBF_* (命令格式字节字符串过滤器参数结构)

| 表 60: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFBF_STRUC_LENGTH_FIXED | 20 | X'00000014' |

MQCFBS_* (命令格式字节字符串参数结构)

| 表 61: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFBS_STRUC_LENGTH_FIXED | 16 | X'00000010' |

MQCF*_* (命令格式头控制选项)

| 表 62: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCF*_LAST | 1 | X'00000001' |
| MQCF*_NOT_LAST | 0 | X'00000000' |

MQCFGR_* (命令格式组参数结构)

| 表 63: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFGR_STRUC_LENGTH | 16 | X'00000010' |

MQCFH_* (命令格式头结构)

| 表 64: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式整数过滤器参数结构)

| 表 65: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFIF_STRUC_LENGTH | 20 | X'00000014' |

MQCFIL_* (命令格式整数列表参数结构)

| 表 66: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFIL_STRUC_LENGTH_FIXED | 16 | X'00000010' |

MQCFIL64_* (命令格式 64 位整数列表参数结构)

| 表 67: 常量值 | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFIL64_STRUC_LENGTH_FIXED | 16 | X'00000010' |

MQCFIN_* (命令格式整数参数结构)

| 表 68: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFIN_STRUC_LENGTH | 16 | X'00000010' |

MQCFIN64_* (命令格式 64 位整数参数结构)

| 表 69: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFIN64_STRUC_LENGTH | 24 | X'00000018' |

MQCFO_* (命令格式刷新存储库选项和命令格式移除队列选项)

命令格式刷新存储库选项

| 表 70: 常量值 | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFO_REFRESH_REPOSITORY_YES | 1 | X'00000001' |
| MQCFO_REFRESH_REPOSITORY_NO | 0 | X'00000000' |

命令格式除去队列选项

| 表 71: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFO_REMOVE_QUEUES_YES | 1 | X'00000001' |
| MQCFO_REMOVE_QUEUES_NO | 0 | X'00000000' |

MQCFOP_* (命令格式过滤器运算符)

| 表 72: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFOP_LESS | 1 | X'00000001' |
| MQCFOP_EQUAL | 2 | X'00000002' |
| MQCFOP_更大 | 4 | X'00000004' |
| MQCFOP_NOT_LESS | 6 | X'00000006' |
| MQCFOP_NOT_EQUAL | 5 | X'00000005' |

| 表 72: 常量值 (继续) | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFOP_NOT_更大 | 3 | X'00000003' |
| MQCFOP_LIKE | 18 | X'00000012' |
| MQCFOP_NOT_LIKE | 21 | X'00000015' |
| MQCFOP_CONTAINS | 10 | X'0000000A' |
| MQCFOP_排除 | 13 | X'0000000D' |
| MQCFOP_CONTAINS_GEN | 26 | X'0000001A' |
| MQCFOP_EXCLUDES_GEN | 29 | X'0000001D' |

MQCFR_* (CF 可恢复性)

| 表 73: 常量值 | | |
|-----------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFR_YES | 1 | X'00000001' |
| MQCFR_NO | 0 | X'00000000' |

MQCFSF_* (命令格式字符串过滤器参数结构)

| 表 74: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFSF_STRUC_LENGTH_FIXED | 24 | X'00000018' |

MQCFSL_* (命令格式字符串列表参数结构)

| 表 75: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFSL_STRUC_LENGTH_FIXED | 24 | X'00000018' |

MQCFST_* (命令格式字符串参数结构)

| 表 76: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFST_STRUC_LENGTH_FIXED | 20 | X'00000014' |

MQCFSTATUS_* (命令格式 CF 状态)

| 表 77: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |
| MQCFSTATUS_ADMIN_不完整 | 20 | X'00000014' |

| 表 77: 常量值 (继续) | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFSTATUS_NEVER_USED | 21 | X'00000015' |
| MQCFSTATUS_NO_BACKUP | 22 | X'00000016' |
| MQCFSTATUS_NOT_FAILED | 23 | X'00000017' |
| MQCFSTATUS_NOT_可恢复 | 24 | X'00000018' |
| MQCFSTATUS_XES_ERROR | 25 | X'00000019' |

MQCFT_* (命令格式的结构类型)

| 表 78: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_记帐 | 22 | X'00000016' |
| MQCFT_INTEGER64 | 23 | X'00000017' |
| MQCFT_INTEGER64_LIST | 25 | X'00000019' |

MQCFATYPE_* (命令格式 CF 类型)

| 表 79: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCFATYPE_APPL | 0 | X'00000000' |
| MQCFATYPE_ADMIN | 1 | X'00000001' |

MQCFUNC_* (CICS 信息头函数)

| 表 80: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| 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 | '↵','↵','↵','↵' |

注: 符号 ↵ 表示单个空白字符。

MQCGWI_* (CICS 信息头获取等待时间间隔)

| 表 81: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCGWI_DEFAULT | -2 | X'FFFFFFFE' |

MQCHAD_* (通道自动定义)

| 表 82: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHAD_DISABLED | 0 | X'00000000' |
| MQCHAD_ENABLED | 1 | X'00000001' |

MQCHIDS_* (命令格式不确定状态)

| 表 83: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHIDS_NOT_INDOUBT | 0 | X'00000000' |
| MQCHIDS_INDOUBT | 1 | X'00000001' |

MQCHLD_* (命令格式通道处置)

| 表 84: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHLD_ALL | -1 | X'FFFFFFF' |
| MQCHLD_DEFAULT | 1 | X'00000001' |

| 表 84: 常量值 (继续) | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHLD_SHARED | 2 | X'00000002' |
| MQCHLD_PRIVATE | 4 | X'00000004' |
| MQCHLD_FIXSHARED | 5 | X'00000005' |

MQCHS_* (命令格式通道状态)

| 表 85: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式通道共享重新启动选项)

| 表 86: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHSH_RESTART_NO | 0 | X'00000000' |
| MQCHSH_RESTART_YES | 1 | X'00000001' |

MQCHSR_* (命令格式通道停止选项)

| 表 87: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHSR_STOP_NOT_REQUESTED | 0 | X'00000000' |
| 已请求 MQCHSR_STOP_REQUESTED | 1 | X'00000001' |

MQCHSSTATE_* (命令格式通道子状态)

| 表 88: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHSSTATE_OTHER | 0 | X'00000000' |
| MQCHSSTATE_END_OF_BATCH | 100 | X'00000064' |
| MQCHSSTATE_正在发送 | 200 | X'000000C8' |
| MQCHSSTATE_正在接收 | 300 | X'0000012C' |
| MQCHSSTATE_SERI 正在进行序列化 | 400 | X'00000190' |
| MQCHSSTATE_RESYNCHING | 500 | X'000001F4' |

| 表 88: 常量值 (继续) | | |
|-------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQCHSSTATE_HEART 跳动 | 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_CONNEC 丁 | 1250 | X'000004E2' |
| MQCHSSTATE_SSL_HAND 震动 | 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_正在压缩 | 1800 | X'00000708' |

MQCHT_* (通道类型)

| 表 89: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式通道表类型)

| 表 90: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCHTAB_Q_MGR | 1 | X'00000001' |
| MQCHTAB_CLNTCONN | 2 | X'00000002' |

MQCI_* (关联标识)

| 表 91: 常量名称和值 | |
|------------------|--------------------------|
| : | 值 |
| MQCI_NONE | X'00...00' (24 个空值) |
| MQCI_NONE_ARRAY | '\0', '\0', ... (24 个空值) |
| MQCI_NEW_SESSION | X'414D5121...' |

| 表 91: 常量名称和值 (继续) | |
|------------------------|------------------------------------|
| : | 值 |
| MQCI_NEW_SESSION_ARRAY | '\x41', '\x4D', '\51', '\x21', ... |

MQCIH_* (CICS 信息头结构和标志)

CICS 信息头结构

| 表 92: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQCIH_STRUC_ID | "CIH↵" |
| MQCIH_STRUC_ID_ARRAY | 'C', 'I', 'H', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 93: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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 信息头标志

| 表 94: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (集群高速缓存类型)

| 表 95: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCLCT_STATIC | 0 | X'00000000' |
| MQCLCT_DYNAMIC | 1 | X'00000001' |

MQCLRS_* (命令格式 "清除主题字符串" 作用域)

| 表 96: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCLRS_LOCAL | 1 | X'00000001' |
| MQCLRS_GLOBAL | 2 | X'00000002' |

MQCLRT_* (命令格式 "清除主题字符串类型")

| 表 97: 常量值 | | |
|-----------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCLRT_留存 | 1 | X'00000001' |

MQCLT_* (CICS 信息头链接类型)

| 表 98: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCLT_PROGRAM | 1 | X'00000001' |
| MQCLT_TRANSACTION | 2 | X'00000002' |

MQCLWL_* (集群工作负载)

| 表 99: 常量值 | | |
|----------------------|-----|--------------|
| : | 小数值 | 十六进制值 |
| MQCLWL_USEQ_LOCAL | 0 | X'00000000' |
| MQCLWL_USEQ_ANY | 1 | X'00000001' |
| MQCLWL_USEQ_AS_Q_MGR | -3 | X'FFFFFFFFD' |

MQCLXQ_* (集群传输队列类型)

MQCLXQ_* 是可以在 DEFCLXQ 队列管理器属性中设置的值。DEFCLXQ 属性 控制缺省情况下集群发送方通道选择从哪个传输队列获取消息， 以将消息发送到集群接收方通道。

| 表 100: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCLXQ_SCTQ | 0 | X'00000000' |
| MQCLXQ_CHANNEL | 1 | X'00000001' |

相关参考

第 716 页的『DefClusterXmitQueue 类型 (MQLONG)』

DefClusterXmitQueue 类型 属性 控制缺省情况下集群发送方通道选择从哪个传输队列获取消息， 以将消息发送到集群接收方通道。

[更改队列管理器](#)

[查询队列管理器](#)

[查询队列管理器 \(响应\)](#)

第 619 页的『MQINQ-查询对象属性』

MQINQ 调用返回整数数组和一组包含对象属性的字符串。

MQCMD_* (命令代码)

| 表 101: 常量值 | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 101: 常量值 (继续) | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 101: 常量值 (继续) | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 101: 常量值 (继续) | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 101: 常量值 (继续) | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式命令信息值)

| 表 102: 常量值 | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 已接受 MQCMDI_CMDScope_接受 | 1 | X'00000001' |
| MQCMDI_CMDScope_GENERATED | 2 | X'00000002' |
| MQCMDI_CMDScope_COMPLETED | 3 | X'00000003' |
| MQCMDI_QSG_DISP_COMPLETED | 4 | X'00000004' |
| 已接受 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_* (命令级别)

| 表 103: 常量名称和值 | |
|------------------|-----|
| : | 值 |
| MQCMDL_LEVEL_1 | 100 |
| MQCMDL_LEVEL_101 | 101 |
| MQCMDL_LEVEL_110 | 110 |

| 表 103: 常量名称和值 (继续) | |
|--------------------|-----|
| : | 值 |
| 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_* (创建消息句柄选项和结构)

创建消息句柄选项结构

| 表 104: 常量结构 | |
|-----------------------|--------------------|
| : | 结构 |
| MQCMHO_STRUC_ID | "CMHO" |
| MQCMHO_STRUC_ID_ARRAY | 'C', 'M', 'H', 'O' |

注: 符号 ↵ 表示单个空白字符。

| 表 105: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCMHO_VERSION_1 | 1 | X'00000001' |
| MQCMHO_CURRENT_VERSION | 1 | X'00000001' |

创建消息句柄选项

| 表 106: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCMHO_DEFAULT_VALIDATION | 0 | X'00000000' |
| MQCMHO_NO_VALIDATION | 1 | X'00000001' |
| MQCMHO_VALIDATE | 2 | X'00000002' |
| MQCMHO_NONE | 0 | X'00000000' |

MQCNO_* (连接选项和结构)

连接选项结构

| 表 107: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQCNO_STRUC_ID | "CNO↵" |
| MQCNO_STRUC_ID_ARRAY | 'C','N','O','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 108: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

连接选项

| 表 109: 常量值 | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 109: 常量值 (继续) | | |
|-------------------------------|-----------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (关闭选项)

| 表 110: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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 信息头输出数据长度)

| 表 111: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCODL_AS_INPUT | -1 | X'FFFFFFFF' |

MQCOMPRESS_* (通道压缩)

| 表 112: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 112: 常量值 (继续) | | |
|-----------------|-----------|-------------|
| : | 小数值 | 十六进制值 |
| MQCOMPRESS_ANY | 268435455 | X'0FFFFFFF' |

MQCONNID_* (连接标识)

| 表 113: 常量名称和值 | |
|---------------------|--------------------------|
| : | 值 |
| MQCONNID_NONE | X'00...00' (24 个空值) |
| MQCONNID_NONE_ARRAY | '\0', '\0', ... (24 个空值) |

MQCOPY_* (属性复制选项)

| 表 114: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCOPY_NONE | 0 | X'00000000' |
| MQCOPY_ALL | 1 | X'00000001' |
| MQCOPY_FOR 何承天 | 2 | X'00000002' |
| MQCOPY_PUBLISH | 4 | X'00000004' |
| MQCOPY_REPLY | 8 | X'00000008' |
| MQCOPY_REPORT | 16 | X'00000010' |
| MQCOPY_DEFAULT | 22 | X'00000016' |

MQCQT_* (集群队列类型)

| 表 115: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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 信息头返回码)

| 表 116: 常量值 | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (MQCBC 常量使用者状态)

| 表 117: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCS_NONE | 0 | X'00000000' |
| MQCS_SUSPENDED_TEMPORARY | 1 | X'00000001' |
| MQCS_SUSPENDED_USER_ACTION | 2 | X'00000002' |
| 已暂挂 MQCS_SUSPENDED | 3 | X'00000003' |
| 已停止 MQCS_STOPPED | 4 | X'00000004' |

MQCSC_* (CICS 信息头启动代码)

| 表 118: 常量结构 | |
|-----------------------|-----------------------|
| : | 结构 |
| 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 | '-', '- ', '- ', '- ' |

注: 符号 - 表示单个空白字符。

MQCSP_* (连接安全性参数结构和认证类型)

连接安全性参数结构

| 表 119: 常量结构 | |
|----------------------|---------------------|
| : | 结构 |
| MQCSP_STRUC_ID | "CSP-" |
| MQCSP_STRUC_ID_ARRAY | 'C', 'S', 'P', '- ' |

注: 符号 - 表示单个空白字符。

| 表 120: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCSP_VERSION_1 | 1 | X'00000001' |
| MQCSP_CURRENT_VERSION | 1 | X'00000001' |

连接安全性参数认证类型

| 表 121: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCSP_AUTH_NONE | 0 | X'00000000' |
| MQCSP_AUTH_USER_ID_AND_PWD | 1 | X'00000001' |

MQCSRV_* (命令服务器选项)

| 表 122: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (队列管理器连接标记)

| 表 123: 常量名称和值 | |
|-----------------|---------------------------|
| : | 值 |
| MQCT_NONE | X'00...00' (128 个空值) |
| MQCT_NONE_ARRAY | '\0', '\0', ... (128 个空值) |

MQCTES_* (CICS 信息头任务结束状态)

| 表 124: 常量值 | | |
|----------------|-------|-------------|
| : | 小数值 | 十六进制值 |
| MQCTES_NOSYNC | 0 | X'00000000' |
| MQCTES_COMMIT | 256 | X'00000100' |
| MQCTES_BACKOUT | 4352 | X'00001100' |
| MQCTES_ENDTASK | 65536 | X'00010000' |

MQCTLO_* (MQCTL 选项结构和使用者控制选项)

MQCTL 选项结构

| 表 125: 常量结构 | |
|-----------------------|--------------------|
| : | 结构 |
| MQCTLO_STRUC_ID | "CTLO" |
| MQCTLO_STRUC_ID_ARRAY | 'C', 'T', 'L', 'O' |

注: 符号 ↵ 表示单个空白字符。

| 表 126: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQCTLO_VERSION_1 | 1 | X'00000001' |
| MQCTLO_CURRENT_VERSION | 1 | X'00000001' |

MQCTL 选项使用者控制选项

| 表 127: 常量值 | | |
|--------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQCTLO_NONE | 0 | X'00000000' |
| MQCTLO_THREAD_AFFINITY | 1 | X'00000001' |
| MQCTLO_FAIL_IF QUIESCING | 8192 | X'00002000' |

MQCUOWC_* (CICS 信息头工作单元控制)

| 表 128: 常量值 | | |
|------------------|-------|-------------|
| : | 小数值 | 十六进制值 |
| MQCUOWC_ONLY | 273 | X'00000111' |
| MQCUOWC_CONTINUE | 65536 | X'00010000' |
| MQCUOWC_FIRST | 17 | X'00000011' |
| MQCUOWC_中间件 | 16 | X'00000010' |
| MQCUOWC_LAST | 272 | X'00000110' |
| MQCUOWC_COMMIT | 256 | X'00000100' |
| MQCUOWC_BACKOUT | 4352 | X'00001100' |

MQCXP_* (通道出口参数结构)

| 表 129: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQCXP_STRUC_ID | "CXP↵" |
| MQCXP_STRUC_ID_ARRAY | 'C', 'X', 'P', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 130: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (目标类)

| 表 131: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDC_MANAGED | 1 | X'00000001' |
| MQDC_PROVIDED | 2 | X'00000002' |

MQDCC_* (转换选项以及掩码和因子)

转换选项

| 表 132: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDCC_DEFAULT_CONVERSION | 1 | X'00000001' |

| 表 132: 常量值 (继续) | | |
|------------------------------|--|-------------|
| : | 小数值 | 十六进制值 |
| 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_逆向 | 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_逆向 | 512 | X'00000200' |
| MQDCC_TARGET_ENC_UNDEFINED | 0 | X'00000000' |
| MQDCC_NONE | 0 | X'00000000' |

转换选项掩码和因子

| 表 133: 常量值 | | |
|-------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (发布/预订删除选项)

| 表 134: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDELO_NONE | 0 | X'00000000' |
| MQDELO_LOCAL | 4 | X'00000004' |

MQDH_* (分发头结构)

| 表 135: 常量结构 | |
|---------------------|--------------------|
| : | 结构 |
| MQDH_STRUC_ID | "DH↵↵" |
| MQDH_STRUC_ID_ARRAY | 'D', 'H', '↵', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 136: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDH_VERSION_1 | 1 | X'00000001' |
| MQDH_CURRENT_VERSION | 1 | X'00000001' |

MQDHF_* (分发头标志)

| 表 137: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDHF_NEW_MSG_IDS | 1 | X'00000001' |
| MQDHF_NONE | 0 | X'00000000' |

MQDISCONNECT_* (命令格式断开连接类型)

| 表 138: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDISCONNECT_NORMAL | 0 | X'00000000' |
| MQDISCONNECT_IMPLICIT | 1 | X'00000001' |
| MQDISCONNECT_Q_MGR | 2 | X'00000002' |

MQDL_* (分发列表)

| 表 139: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDL_SUPPORTED | 1 | X'00000001' |
| MQDL_NOT_SUPPORTED | 0 | X'00000000' |

MQDLH_* (死信头结构)

| 表 140: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQDLH_STRUC_ID | "DLH↵" |
| MQDLH_STRUC_ID_ARRAY | 'D','L','H','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 141: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDLH_VERSION_1 | 1 | X'00000001' |
| MQDLH_CURRENT_VERSION | 1 | X'00000001' |

MQDLV_* (持久/非持久消息传递)

| 表 142: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDLV_AS_PARENT | 0 | X'00000000' |
| MQDLV_ALL | 1 | X'00000001' |
| MQDLV_ALL_DUR | 2 | X'00000002' |
| MQDLV_ALL_AVAIL | 3 | X'00000003' |

MQDMHO_* (删除消息句柄选项和结构)

删除消息句柄选项结构

| 表 143: 常量结构 | |
|-----------------------|-----------------|
| : | 结构 |
| MQDMHO_STRUC_ID | "DMHO" |
| MQDMHO_STRUC_ID_ARRAY | 'D','M','H','O' |

注: 符号 - 表示单个空白字符。

| 表 144: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDMHO_VERSION_1 | 1 | X'00000001' |
| MQDMHO_CURRENT_VERSION | 1 | X'00000001' |

删除消息句柄选项

| 表 145: 常量值 | | |
|-------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDMHO_NONE | 0 | X'00000000' |

MQDMPO_* (删除消息属性选项和结构)

删除消息属性选项结构

| 表 146: 常量结构 | |
|-----------------------|-----------------|
| : | 结构 |
| MQDMPO_STRUC_ID | "DMPO" |
| MQDMPO_STRUC_ID_ARRAY | 'D','M','P','O' |

注: 符号 - 表示单个空白字符。

| 表 147: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDMPO_VERSION_1 | 1 | X'00000001' |
| MQDMPO_CURRENT_VERSION | 1 | X'00000001' |

删除消息属性选项

| 表 148: 常量值 | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDMPO_DEL_FIRST | 0 | X'00000000' |
| MQDMPO_DEL_PROP_UNDER_CURSOR | 1 | X'00000001' |
| MQDMPO_NONE | 0 | X'00000000' |

MQDNSWLM_* (DNS WLM)

| 表 149: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDNSWLM_NO | 0 | X'00000000' |
| MQDNSWLM_YES | 1 | X'00000001' |

MQDT_* (目标类型)

| 表 150: 常量值 | | |
|-------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDT_APPL | 1 | X'00000001' |
| MQDT_BROKER | 2 | X'00000002' |

MQDXP_* (转换出口参数结构)

| 表 151: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQDXP_STRUC_ID | "DXP↵" |
| MQDXP_STRUC_ID_ARRAY | 'D','X','P','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 152: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQDXP_VERSION_1 | 1 | X'00000001' |
| MQDXP_VERSION_2 | 2 | X'00000002' |
| MQDXP_CURRENT_VERSION | 2 | X'00000002' |

MQEC_* (信号值)

| 表 153: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQEC_MSG_已到达 | 2 | X'00000002' |
| MQEC_WAIT_INTERVAL_EXPIRED | 3 | X'00000003' |
| MQEC_WAIT_CANCEL | 4 | X'00000004' |
| MQEC_Q_MGR QUIESCING | 5 | X'00000005' |
| MQEC_CONNECTION QUIESCING | 6 | X'00000006' |

MQEI_* (到期)

| 表 154: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQEI_UNLIMITED | -1 | X'FFFFFFFF' |

MQENC_* (编码)

MQENC_* (编码)

| : | 平台 | 小数值 | 十六进制值 |
|--------------|------------------------------|-------------|-------------|
| MQENC_NATIVE | IBM i | 273 | X'00000111' |
| | Linux | 546 | X'00000222' |
| | SPARC 上的 Linux | 273 | X'00000111' |
| | x86 上的 Linux | 546 | X'00000222' |
| | SPARC 上的 Solaris | 273 | X'00000111' |
| | UNIX 系统 | 273 | X'00000111' |
| | Windows | 546 | X'00000222' |
| | Windows 上的 Micro Focus COBOL | 17 | X'00000011' |
| z/OS | 785 | X'00000311' | |

MQENC_* (编码掩码)

| : | 小数值 | 十六进制值 |
|---------------------|-------|-------------|
| MQENC_INTEGER_MASK | 15 | X'0000000F' |
| MQENC_DECIMAL_MASK | 240 | X'000000F0' |
| MQENC_FLOAT_MASK | 3840 | X'00000F00' |
| MQENC_RESERVED_MASK | -4096 | X'FFFFFF00' |

MQENC_* (二进制整数的编码)

| : | 小数值 | 十六进制值 |
|-------------------------|-----|-------------|
| MQENC_INTEGER_UNDEFINED | 0 | X'00000000' |
| MQENC_INTEGER_NORMAL | 1 | X'00000001' |
| MQENC_INTEGER_REVERSED | 2 | X'00000002' |

MQENC_* (压缩十进制整数的编码)

| : | 小数值 | 十六进制值 |
|-------------------------|-----|-------------|
| MQENC_DECIMAL_UNDEFINED | 0 | X'00000000' |
| MQENC_DECIMAL_NORMAL | 16 | X'00000010' |
| MQENC_DECIMAL_REVERSED | 32 | X'00000020' |

MQENC_* (浮点数编码)

| : | 小数值 | 十六进制值 |
|-----------------------|-----|-------------|
| MQENC_FLOAT_UNDEFINED | 0 | X'00000000' |

| 表 159: 常量值 (继续) | | |
|---------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (嵌入式命令格式头结构和标志)

嵌入式命令格式头结构

| 表 160: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQEPH_STRUC_ID | "EPH↵" |
| MQEPH_STRUC_ID_ARRAY | 'E', 'P', 'H', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 161: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQEPH_STRUC_LENGTH_FIXED | 68 | X'00000044' |
| MQEPH_VERSION_1 | 1 | X'00000001' |
| MQEPH_CURRENT_VERSION | 1 | X'00000001' |

嵌入式命令格式头标志

| 表 162: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQEPH_NONE | 0 | X'00000000' |
| MQEPH_CCSID_EMBEDDED | 1 | X'00000001' |

MQET_* (命令格式 Escape 类型)

| 表 163: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQET_MQSC | 1 | X'00000001' |

MQEVO_* (命令格式事件源)

| 表 164: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式事件记录)

| 表 165: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQEVR_DISABLED | 0 | X'00000000' |
| MQEVR_ENABLED | 1 | X'00000001' |
| MQEVR_EXCEPTION | 2 | X'00000002' |
| MQEVR_NO_DISPLAY | 3 | X'00000003' |

MQEXPI_* (到期扫描时间间隔)

| 表 166: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQEXPI_OFF | 0 | X'00000000' |

MQFB_* (反馈值)

| 表 167: 常量值 | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_ACTIVactivities | 282 | X'0000011A' |
| MQFB_NOT_FORWARD | 283 | X'0000011B' |
| MQFB_NOT_交付 | 284 | X'0000011C' |
| MQFB_UNSUPPORTED_转发 | 285 | X'0000011D' |

| 表 167: 常量值 (继续) | | |
|-------------------------------|----------|-------------|
| : | 小数值 | 十六进制值 |
| MQFB_UNSUPPORTED_DELIVERY | 286 | X'0000011E' |
| MQFB_DATA_LENGTH_ZERO | 291 | X'00000123' |
| MQFB_DATA_LENGTH_负数 | 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_IM | 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_AB 已结束 | 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_MATCH | 503 | X'000001F7' |
| MQFB_SELECTOR_MATCH | 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 | 99999999 | X'3B9AC9FF' |

MQFC_* (命令格式强制选项)

| 表 168: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQFC_YES | 1 | X'00000001' |
| MQFC_NO | 0 | X'00000000' |

MQFMT_* (格式)

| 表 169: 常量名称和值 | |
|--------------------------------|-------------------------------------|
| : | 值 |
| MQFMT_NONE | "rrrrrrrr" |
| MQFMT_ADMIN | "MQADMINr" |
| MQFMT_CHANNEL_COMPLETED | "MQCHCOMr" |
| MQFMT_CICS | "MQCICSrr" |
| MQFMT_COMMAND_1 | "MQCMD1rr" |
| MQFMT_COMMAND_2 | "MQCMD2rr" |
| MQFMT_DEAD_LETTER_HEADER | "MQDEADrr" |
| MQFMT_DIST_HEADER | "MQHDISTr" |
| MQFMT_EMBEDDED_PCF | "MQHEPCFr" |
| MQFMT_EVENT | "MQEVENTr" |
| MQFMT_IMS | "MQIMSrrr" |
| MQFMT_IMS_VAR_STRING | "MQIMSVSr" |
| MQFMT_MD_EXTENSION | "MQHMDErr" |
| MQFMT_PCF | "MQPCFrrr" |
| MQFMT_REF_MSG_HEADER | "MQHREFrr" |
| MQFMT_RF_HEADER | "MQHRFrrr" |
| MQFMT_RF_HEADER_1 | "MQHRFrrr" |
| MQFMT_RF_HEADER_2 | "MQHRF2rr" |
| MQFMT_STRING | "MQSTRrrr" |
| MQFMT_TRIGGER | "MQTRIGrr" |
| MQFMT_WORK_INFO_HEADER | "MQHWIHRr" |
| MQFMT_XMIT_Q_HEADER | "MQXMITrr" |
| MQFMT_NONE_ARRAY | 'r','r','r','r','r','r','r','r','r' |
| MQFMT_ADMIN_ARRAY | 'M','Q','A','D','M','I','N','r' |
| MQFMT_CHANNEL_COMPLETED_ARRAY | 'M','Q','C','H','C','O','M','r' |
| MQFMT_CICS_ARRAY | 'M','Q','C','I','C','S','r','r' |
| MQFMT_COMMAND_1_ARRAY | 'M','Q','C','M','D','1','r','r' |
| MQFMT_COMMAND_2_ARRAY | 'M','Q','C','M','D','2','r','r' |
| MQFMT_DEAD_LETTER_HEADER_ARRAY | 'M','Q','D','E','A','D','r','r' |
| MQFMT_DIST_HEADER_ARRAY | 'M','Q','H','D','I','S','T','r' |
| MQFMT_EMBEDDED_PCF_ARRAY | 'M','Q','H','E','P','C','F','r' |
| MQFMT_EVENT_ARRAY | 'M','Q','E','V','E','N','T','r' |
| MQFMT_IMS_ARRAY | 'M','Q','I','M','S','r','r','r' |
| MQFMT_IMS_VAR_STRING_ARRAY | 'M','Q','I','M','S','V','S','r' |
| MQFMT_MD_EXTENSION_ARRAY | 'M','Q','H','M','D','E','r','r' |
| MQFMT_PCF_ARRAY | 'M','Q','P','C','F','r','r','r' |
| MQFMT_REF_MSG_HEADER_ARRAY | 'M','Q','H','R','E','F','r','r' |
| MQFMT_RF_HEADER_ARRAY | 'M','Q','H','R','F','r','r','r' |

| 表 169: 常量名称和值 (继续) | |
|------------------------------|---------------------------------|
| : | 值 |
| 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',' ',' ' |

注: 符号 ' ' 表示单个空白字符。

MQGA_* (组属性选择器)

| 表 170: 常量值 | | | |
|------------|--|------|-------------|
| : | | 小数值 | 十六进制值 |
| MQGA_FIRST | | 8001 | X'00001F41' |
| MQGA_LAST | | 9000 | X'00002328' |

MQGACF_* (命令格式组参数类型)

| 表 171: 常量值 | | | |
|----------------------------|--|------|-------------|
| : | | 小数值 | 十六进制值 |
| 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_* (组标识)

| 表 172: 常量名称和值 | |
|-----------------|--------------------------|
| : | 值 |
| MQGI_NONE | X'00...00' (24 个空值) |
| MQGI_NONE_ARRAY | '\0', '\0', ... (24 个空值) |

MQGMO_* (获取消息选项和结构)

获取消息选项结构

| : | 结构 |
|----------------------|-----------------|
| MQGMO_STRUC_ID | "GMO↵" |
| MQGMO_STRUC_ID_ARRAY | 'G','M','O','↵' |

注: 符号 ↵ 表示单个空白字符。

| : | 小数值 | 十六进制值 |
|-----------------------|-----|-------------|
| 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' |

获取消息选项

| : | 小数值 | 十六进制值 |
|-------------------------------|----------|-------------|
| 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'00008000' |
| 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' |

| 表 175: 常量值 (继续) | | |
|--------------------------------|-----------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (组状态)

| 表 176: 常量名称和值 | |
|------------------------|-----|
| : | 值 |
| MQGS_NOT_IN_GROUP | '-' |
| MQGS_MSG_IN_GROUP | 'G' |
| MQGS_LAST_MSG_IN_GROUP | 'L' |

注: 符号 - 表示单个空白字符。

MQHA_* (句柄选择器)

| 表 177: 常量值 | | |
|-----------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQHA_FIRST | 4001 | X'00000FA1' |
| MQHA_BAG_HANDLE | 4001 | X'00000FA1' |
| MQHA_LAST_USED | 4001 | X'00000FA1' |
| MQHA_LAST | 6000 | X'00001770' |

MQHB_* (包句柄)

| 表 178: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQHB_UNUSABLE_HBAG | -1 | X'FFFFFFFF' |
| MQHB_NONE | -2 | X'FFFFFFFE' |

MQHC_* (连接句柄)

| 表 179: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQHC_DEF_HCONN | 0 | X'00000000' |
| MQHC_UNUSABLE_HCONN | -1 | X'FFFFFFFF' |

| 表 179: 常量值 (继续) | | |
|-------------------------|-----|--------------|
| : | 小数值 | 十六进制值 |
| MQHC_UNASSOCIATED_HCONN | -3 | X'FFFFFFFFD' |

MQHM_* (消息句柄)

| 表 180: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQHM_UNUSABLE_HMSG | -1 | X'FFFFFFFF' |
| MQHM_NONE | 0 | X'00000000' |

MQHO_* (对象句柄)

| 表 181: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQHO_UNUSABLE_HOBJ | -1 | X'FFFFFFFF' |
| MQHO_NONE | 0 | X'00000000' |

MQHSTATE_* (命令格式句柄状态)

| 表 182: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQHSTATE_INACTIVE | 0 | X'00000000' |
| MQHSTATE_ACTIVE | 1 | X'00000001' |

MQIA_* (整数属性选择器)

| 表 183: 常量值 | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 183: 常量值 (继续) | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 183: 常量值 (继续) | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_QUEUEING | 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' |

| 表 183: 常量值 (继续) | | |
|--------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 183: 常量值 (继续) | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 183: 常量值 (继续) | | |
|----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式整数参数类型)

| 表 184: 常量值 | | |
|-----------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 184: 常量值 (继续) | | |
|------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_IDENTIFIE | 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_查询 | 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' |

| 表 184: 常量值 (继续) | | |
|-----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_QUEUE | 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' |

| 表 184: 常量值 (继续) | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_个扩展数据块 | 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' |

| 表 184: 常量值 (继续) | | |
|------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_DELOC_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' |

| 表 184: 常量值 (继续) | | |
|---------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_ACTIVactivities | 1235 | X'000004D3' |
| MQIACF_MAX_ACTIVactivities | 1236 | X'000004D4' |
| MQIACF_DISCONTINUITY_COUNT | 1237 | X'000004D5' |
| Mqiacf_route_累加 | 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' |

| 表 184: 常量值 (继续) | | |
|-----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_ACTIV 息 | 1257 | X'000004E9' |
| MQIACF_MONITORING | 1258 | X'000004EA' |
| MQIACF_ROUTE_转发 | 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' |

| 表 184: 常量值 (继续) | | |
|----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式整数通道类型)

| 表 185: 常量值 | | |
|-----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 185: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 185: 常量值 (继续) | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 185: 常量值 (继续) | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式整数监视参数类型)

| 表 186: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 186: 常量值 (继续) | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_批处理 | 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_批处理 | 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' |

| 表 186: 常量值 (继续) | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_PUBLIC_ISH_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' |

| 表 186: 常量值 (继续) | | |
|--------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_交付 | 819 | X'00000333' |
| MQIAMO_PKTS_PROCESSED | 820 | X'00000334' |
| MQIAMO_PKTS_DLVD | 821 | X'00000335' |
| MQIAMO_PKTS_SENT | 822 | X'00000336' |
| MQIAMO_PKTS_重复 | 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_PROCESSED | 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' |

| 表 186: 常量值 (继续) | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIAMO_TOTAL_MSGS_SELECTED | 834 | X'00000342' |
| MQIAMO_TOTAL_MSGS_EXPIRED | 835 | X'00000343' |
| MQIAMO_TOTAL_MSGS_交付 | 836 | X'00000344' |
| MQIAMO_TOTAL_MSGS_退回 | 837 | X'00000345' |
| MQIAMO_LAST_USED | 837 | X'00000345' |

MQIAMO64_* (命令格式 64 位整数监视参数类型)

| 表 187: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (整数系统选择器)

| 表 188: 常量值 | | |
|--------------------------|-------|--------------|
| : | 小数值 | 十六进制值 |
| 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'FFFFFF830' |

MQIAUT_* (IMS 信息头认证程序)

| 表 189: 常量名称和值 | |
|-------------------|---|
| : | 值 |
| MQIAUT_NONE | "rrrrrrrrrr" |
| MQIAUT_NONE_ARRAY | 'r','r','r','r','r','r','r','r','r','r','r','r','r','r','r','r' |

注: 符号 r 表示单个空白字符。

MQIAV_* (整数属性值)

| 表 190: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIAV_NOT_APPLICABLE | -1 | X'FFFFFFFF' |
| MQIAV_UNDEFINED | -2 | X'FFFFFFFE' |

MQICM_* (IMS 信息头落实方式)

| 表 191: 常量名称和值 | |
|------------------------|-----|
| : | 值 |
| MQICM_COMMIT_THEN_SEND | '0' |
| MQICM_SEND_THEN_COMMIT | '1' |

MQIDO_* (命令格式不确定选项)

| 表 192: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIDO_COMMIT | 1 | X'00000001' |
| MQIDO_BACKOUT | 2 | X'00000002' |

MQIEP_* (接口入口点)

连接安全性参数结构

| 表 193: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQIEP_STRUC_ID | "IEP r" |
| MQIEP_STRUC_ID_ARRAY | 'I','E','P','r' |

注: 符号 r 表示单个空白字符。

| 表 194: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIEP_VERSION_1 | 1 | X'00000001' |
| MQDXP_CURRENT_VERSION | 1 | X'00000001' |

MQIGQ_* (组内排队)

| 表 195: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIGQ_DISABLED | 0 | X'00000000' |
| MQIGQ_ENABLED | 1 | X'00000001' |

MQIGQPA_* (组内排队放置权限)

| 表 196: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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 信息头结构和标志)

IMS 信息头结构

| 表 197: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQIIH_STRUC_ID | "IIH↵" |
| MQIIH_STRUC_ID_ARRAY | 'I','I','H','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 198: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIIH_VERSION_1 | 1 | X'00000001' |
| MQIIH_CURRENT_VERSION | 1 | X'00000001' |
| MQIIH_LENGTH_1 | 84 | X'00000054' |

IMS 信息头标志

| 表 199: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (查询消息属性选项和结构)

查询消息属性选项结构

| 表 200: 常量结构 | |
|-----------------------|-----------------|
| : | 结构 |
| MQIMPO_STRUC_ID | "IMPO" |
| MQIMPO_STRUC_ID_ARRAY | 'I','M','P','O' |

注: 符号 - 表示单个空白字符。

| 表 201: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIMPO_VERSION_1 | 1 | X'00000001' |
| MQIMPO_CURRENT_VERSION | 1 | X'00000001' |

查询消息属性选项

| 表 202: 常量值 | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式进站处置)

| 表 203: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQINBD_Q_MGR | 0 | X'00000000' |
| MQINBD_GROUP | 3 | X'00000003' |

MQIND_* (特殊索引值)

| 表 204: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIND_NONE | -1 | X'FFFFFFFF' |
| MQIND_ALL | -2 | X'FFFFFFFE' |

MQIPADDR_* (IP 地址版本)

| 表 205: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQIPADDR_IPV4 | 0 | X'00000000' |
| MQIPADDR_IPV6 | 1 | X'00000001' |

MQISS_* (IMS 信息头安全作用域)

| 表 206: 常量名称和值 | |
|---------------|-----|
| : | 值 |
| MQISS_CHECK | 'C' |
| MQISS_FULL | 'F' |

MQIT_* (索引类型)

| 表 207: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (mqInquireItemInfo 的项类型)

| 表 208: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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 信息头事务实例标识)

| 表 209: 常量名称和值 | |
|-------------------|--------------------------|
| : | 值 |
| MQITII_NONE | X'00...00' (16 个空值) |
| MQITII_NONE_ARRAY | '\0', '\0', ... (16 个空值) |

MQITS_* (IMS 信息头事务状态)

| 表 210: 常量名称和值 | |
|---------------------------|-----|
| : | 值 |
| MQITS_IN_CONVERSATION | 'C' |
| MQITS_NOT_IN_CONVERSATION | '-' |
| MQITS_ARCHITECTED | 'A' |

注: 符号 - 表示单个空白字符。

MQKAI_* (KeepAlive 时间间隔)

| 表 211: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQKAI_AUTO | -1 | X'FFFFFFFF' |

MQMASTER_* (主管理)

| 表 212: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMASTER_NO | 0 | X'00000000' |
| MQMASTER_YES | 1 | X'00000001' |

MQMCAS_* (命令格式消息通道代理程序状态)

| 表 213: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMCAS_STOPPED | 0 | X'00000000' |
| MQMCAS_RUNNING | 3 | X'00000003' |

MQMCAT_* (MCA 类型)

| 表 214: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMCAT_PROCESS | 1 | X'00000001' |
| MQMCAT_THREAD | 2 | X'00000002' |

MQMCD_* (发布/预订选项标记信息)

发布/预订选项标记消息内容描述符 (mcd) 标记

| 表 215: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMCD_FOLDER_VERSION | 1 | X'00000001' |

发布/预订选项标记名称

| 表 216: 常量名称和值 | |
|------------------|--------|
| : | 值 |
| MQMCD_MSG_DOMAIN | "Msd" |
| MQMCD_MSG_SET | "Set" |
| MQMCD_MSG_TYPE | "Type" |
| MQMCD_MSG_FORMAT | "Fmt" |

发布/预订选项标记 XML 标记名称

| 表 217: 常量名称和值 | |
|--------------------|-----------|
| : | 值 |
| 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>" |

发布/预订选项标记标记值

| 表 218: 常量名称和值 | |
|-------------------------|--------------|
| : | 值 |
| 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_* (消息描述符结构)

| 表 219: 常量结构 | |
|---------------------|--------------------|
| : | 结构 |
| MQMD_STRUC_ID | "MD↵↵" |
| MQMD_STRUC_ID_ARRAY | 'M', 'D', '↵', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 220: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMD_VERSION_1 | 1 | X'00000001' |
| MQMD_VERSION_2 | 2 | X'00000002' |
| MQMD_CURRENT_VERSION | 2 | X'00000002' |

MQMDE_* (消息描述符扩展结构)

| 表 221: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQMDE_STRUC_ID | "MDE↵" |
| MQMDE_STRUC_ID_ARRAY | 'M', 'D', 'E', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 222: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMDE_VERSION_2 | 2 | X'00000002' |
| MQMDE_CURRENT_VERSION | 2 | X'00000002' |
| MQMDE_LENGTH_2 | 72 | X'00000048' |

MQMDEF_* (消息描述符扩展标志)

| 表 223: 常量值 | | |
|-------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMDEF_NONE | 0 | X'00000000' |

MQMDS_* (消息传递序列)

| 表 224: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMDS_PRIORITY | 0 | X'00000000' |
| MQMDS_FIFO | 1 | X'00000001' |

MQMF_* (消息标志)

| 表 225: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMF_SEGMENTATION_ALLOWED | 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_* (缓冲区选项和结构的消息句柄)

消息句柄到缓冲区选项结构

| 表 226: 常量结构 | |
|-----------------------|--------------------|
| : | 结构 |
| MQMHBO_STRUC_ID | "MHBO" |
| MQMHBO_STRUC_ID_ARRAY | 'M', 'H', 'B', 'O' |

注: 符号 – 表示单个空白字符。

| 表 227: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMHBO_VERSION_1 | 1 | X'00000001' |
| MQMHBO_CURRENT_VERSION | 1 | X'00000001' |

消息句柄到缓冲区选项

| 表 228: 常量值 | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMHBO_PROPERTIES_IN_MQRFH2 | 1 | X'00000001' |
| MQMHBO_DELETE_PROPERTIES | 2 | X'00000002' |
| MQMHBO_NONE | 0 | X'00000000' |

MQMI_* (消息标识)

| 表 229: 常量名称和值 | |
|-----------------|--------------------------|
| : | 值 |
| MQMI_NONE | X'00...00' (24 个空值) |
| MQMI_NONE_ARRAY | '\0', '\0', ... (24 个空值) |

MQMMBI_* (消息标记-浏览时间间隔)

| 表 230: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMMBI_UNLIMITED | -1 | X'FFFFFFFF' |

MQMO_* (匹配选项)

| 表 231: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式方式选项)

| 表 232: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQMODE_FORCE | 0 | X'00000000' |
| MQMODE QUIESCE | 1 | X'00000001' |
| MQMODE_TERMINATE | 2 | X'00000002' |

MQMON_* (监视值)

| 表 233: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (消息类型)

| 表 234: 常量值 | | |
|--------------------------|----------|-------------|
| : | 小数值 | 十六进制值 |
| 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 | 99999999 | X'3B9AC9FF' |

MQMTOK_* (消息令牌)

| 表 235: 常量名称和值 | |
|-------------------|--------------------------|
| : | 值 |
| MQMTOK_NONE | X'00...00' (16 个空值) |
| MQMTOK_NONE_ARRAY | '\0', '\0', ... (16 个空值) |

MQNC_* (名称计数)

| 表 236: 常量值 | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQNC_MAX_NAMELIST_NAME_COUNT | 256 | X'00000100' |

MQNPM_* (非持久消息类)

| 表 237: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQNPM_CLASS_NORMAL | 0 | X'00000000' |
| MQNPM_CLASS_HIGH | 10 | X'0000000A' |

MQNPMS_* (NonPersistent-消息速度)

| 表 238: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQNPMS_NORMAL | 1 | X'00000001' |
| MQNPMS_FAST | 2 | X'00000002' |

MQNT_* (名称列表类型)

| 表 239: 常量值 | | |
|----------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (名称/值字符串的名称)

| 表 240: 常量名称和值 | |
|-----------------|-----------------|
| : | 值 |
| MQNVS_APPL_TYPE | "OPT_APP_GRP↵" |
| MQNVS_MSG_TYPE | "OPT_MSG_TYPE↵" |

注: 符号 ↵ 表示单个空白字符。

MQOA_* (对象属性的选择器限制)

| 表 241: 常量值 | | |
|------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQOA_FIRST | 1 | X'00000001' |
| MQOA_LAST | 9000 | X'00002328' |

MQOD_* (对象描述符结构)

| 表 242: 常量结构 | |
|---------------------|-----------------|
| : | 结构 |
| MQOD_STRUC_ID | "OD↵↵" |
| MQOD_STRUC_ID_ARRAY | '0','D','↵','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 243: 常量值 | | |
|----------------------|--|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (对象实例标识)

| 表 244: 常量名称和值 | |
|------------------|--------------------------|
| : | 值 |
| MQOII_NONE | X'00...00' (24 个空值) |
| MQOII_NONE_ARRAY | '\0', '\0', ... (24 个空值) |

MQOL_* (原始长度)

| 表 245: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQOL_UNDEFINED | -1 | X'FFFFFFFF' |

MQOM_* ("查询组" 上的 Db2 消息选项已过时)

| 表 246: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQOM_NO | 0 | X'00000000' |
| MQOM_YES | 1 | X'00000001' |

MQOO_* (打开选项)

| 表 247: 常量值 | | |
|----------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 247: 常量值 (继续) | | |
|-------------------------------|---------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (以下仅在 C++ 中使用)

| 表 248: 常量值 | | |
|----------------------|--------|-------------|
| : | 小数值 | 十六进制值 |
| MQOO_RESOLVE_NAMES | 65536 | X'00010000' |
| MQOO_RESOLVE_LOCAL_Q | 262144 | X'00040000' |

MQOP_* (MQCTL 和 MQCB 的操作码)

MQCTL 的操作码

| 表 249: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQOP_START | 1 | X'00000001' |
| MQOP_START_WAIT | 2 | X'00000002' |
| MQOP_STOP | 4 | X'00000004' |

MQCB 的操作码

| 表 250: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQOP_REGISTER | 256 | X'00000100' |
| MQOP_DEREGISTER | 512 | X'00000200' |

MQCTL 和 MQCB 的操作码

| 表 251: 常量值 | | |
|--------------|--------|-------------|
| : | 小数值 | 十六进制值 |
| MQOP_SUSPEND | 65536 | X'00010000' |
| MQOP_RESUME | 131072 | X'00020000' |

MQOPEN_* (与 MQOPEN_PRIV 结构相关的值)

| : | 小数值 | 十六进制值 |
|-----------------------------|-----|-------------|
| MQOPEN_PRIV_VERSION_1 | 1 | X'00000001' |
| MQOPEN_PRIV_CURRENT_VERSION | 1 | X'00000001' |

MQOPER_* (活动操作)

| : | 小数值 | 十六进制值 |
|--------------------------|----------|-------------|
| 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 | 99999999 | X'3B9AC9FF' |

MQOT_* (对象类型和扩展对象类型)

对象类型

| : | 小数值 | 十六进制值 |
|--------------------|-----|-------------|
| 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' |

| 表 254: 常量值 (继续) | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQOT_CF_STRUC | 10 | X'0000000A' |
| MQOT_LISTENER | 11 | X'0000000B' |
| MQOT_服务 | 12 | X'0000000C' |
| MQOT_RESERVED_1 | 999 | X'000003E7' |

扩展对象类型

| 表 255: 常量值 | | |
|------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (放置权限)

| 表 256: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPA_DEFAULT | 1 | X'00000001' |
| MQPA_CONTEXT | 2 | X'00000002' |
| MQPA_ONLY_MCA | 3 | X'00000003' |
| MQPA_ALTERNATE_OR_MCA | 4 | X'00000004' |

MQPD_* (属性描述符, 支持和上下文)

属性描述符结构

| 表 257: 常量结构 | |
|---------------------|-----------------|
| : | 结构 |
| MQPD_STRUC_ID | "PD↵" |
| MQPD_STRUC_ID_ARRAY | 'P','D','↵','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 258: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPD_VERSION_1 | 1 | X'00000001' |
| MQPD_CURRENT_VERSION | 1 | X'00000001' |

注: 符号 `␣` 表示单个空白字符。

属性描述符选项

| 表 259: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPD_NONE | 0 | X'00000000' |

属性支持选项

| 表 260: 常量值 | | |
|--------------------------------|----------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

属性上下文

| 表 261: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPD_NO_CONTEXT | 0 | X'00000000' |
| MQPD_USER_CONTEXT | 1 | X'00000001' |

MQPER_* (持久性值)

| 表 262: 常量值 | | |
|--------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (平台)

| | | |
|------------|---|-------------|
| 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_* (发布掩码的放置消息选项和结构)

Put 消息选项结构

| | |
|----------------------|--------------------|
| : | 结构 |
| MQPMO_STRUC_ID | "PMO↵" |
| MQPMO_STRUC_ID_ARRAY | 'P', 'M', 'O', '↵' |

注: 符号 ↵ 表示单个空白字符。

| : | 小数值 | 十六进制值 |
|-----------------------|--|--|
| 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) |

放置消息 选项

| : | 小数值 | 十六进制值 |
|-----------------------|-----|-------------|
| MQPMO_SYNCPOINT | 2 | X'00000002' |
| MQPMO_NO_SYNCPOINT | 4 | X'00000004' |
| MQPMO_DEFAULT_CONTEXT | 32 | X'00000020' |
| MQPMO_NEW_MSG_ID | 64 | X'00000040' |

| 表 265: 常量值 (继续) | | |
|--------------------------------|-----------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

用于发布掩码的放置消息选项

| 表 266: 常量值 | | |
|------------------------|---------|-------------|
| : | 小数值 | 十六进制值 |
| MQPMO_PUB_OPTIONS_MASK | 2097152 | X'00200000' |

MQPMRF_* (放置消息记录字段)

| 表 267: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式清除选项)

| 表 268: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPO_YES | 1 | X'00000001' |
| MQPO_NO | 0 | X'00000000' |

MQPRI_* (优先级)

| 表 269: 常量值 | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (队列和通道属性控制值以及最大属性长度)

队列和通道属性控制值

| 表 270: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPROP_COMPATIBILITY | 0 | X'00000000' |
| MQPROP_NONE | 1 | X'00000001' |
| MQPROP_ALL | 2 | X'00000002' |
| MQPROP_FORCE_MQRFH2 | 3 | X'00000003' |

最大属性长度

| 表 271: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPROP_UNRESTRICTED_LENGTH | -1 | X'FFFFFFFF' |

MQPRT_* (放置响应值)

| 表 272: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPRT_RESPONSE_AS_PARENT | 0 | X'00000000' |
| MQPRT_SYNC_RESPONSE | 1 | X'00000001' |
| MQPRT_ASYNC_RESPONSE | 2 | X'00000002' |

MQPS_* (发布/预订)

命令格式发布/预订状态

| 表 273: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPS_STATUS_INACTIVE | 0 | X'00000000' |

| 表 273: 常量值 (继续) | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPS_STATUS_STARing | 1 | X'00000001' |
| MQPS_STATUS_正在停止 | 2 | X'00000002' |
| MQPS_STATUS_ACTIVE | 3 | X'00000003' |
| MQPS_STATUS_COMPAT | 4 | X'00000004' |
| MQPS_STATUS_ERROR | 5 | X'00000005' |
| MQPS_STATUS_MODIFIED | 6 | X'00000006' |

以字符串形式发布/预订标记

| 表 274: 常量值 | | |
|-----------------------------|-------------------|-------|
| : | 小数值 | 十六进制值 |
| MQPS_COMMAND | "MQPSCommand" | |
| MQPS_COMP_CODE | "MQPSCompCode" | |
| MQPS_CORREL_ID | "MQPSCorrelId" | |
| MQPS_DELETE_OPTIONS | "MQPSDel0pts" | |
| MQPS_ERROR_ID | "MQPSErrorId" | |
| MQPS_ERROR_POS | "MQPSErrorPos" | |
| MQPS_INTEGER_DATA | "MQPSIntData" | |
| MQPS_PARAMETER_ID | "MQSParmId" | |
| MQPS_PUBLICATION_OPTIONS | "MQSPub0pts" | |
| MQPS_PUBLISH_TIMESTAMP | "MQSPubTime" | |
| MQPS_Q_MGR_NAME | "MQPSQMgrName" | |
| MQPS_Q_NAME | "MQPSQName" | |
| MQPS_REASON | "MQPSReason" | |
| MQPS_REASON_TEXT | "MQPSReasonText" | |
| MQPS_REGISTRATION_OPTIONS | "MQPSReg0pts" | |
| MQPS_SEQUENCE_NUMBER | "MQPSSeqNum" | |
| MQPS_STREAM_NAME | "MQPStreamName" | |
| MQPS_STRING_DATA | "MQPSStringData" | |
| MQPS_SUBSCRIPTION_IDENTITY | "MQPSSubIdentity" | |
| MQPS_SUBSCRIPTION_NAME | "MQPSSubName" | |
| MQPS_SUBSCRIPTION_USER_DATA | "MQPSSubUserData" | |
| MQPS_TOPIC | "MQPSTopic" | |
| MQPS_USER_ID | "MQPSUserId" | |

以空白括起的字符串形式发布/预订标记

| 表 275: 常量值 | | |
|------------------|------------------|-------|
| : | 小数值 | 十六进制值 |
| MQPS_COMMAND_B | "bMQPSCommandb" | |
| MQPS_COMP_CODE_B | "bMQPSCompCodeb" | |

| 表 275: 常量值 (继续) | | |
|-------------------------------|---------------------|-------|
| : | 小数值 | 十六进制值 |
| 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 | "bMQSParmIdb" | |
| MQPS_PUBLICATION_OPTIONS_B | "bMQSPubOptsb" | |
| MQPS_PUBLISH_TIMESTAMP_B | "bMQSPubTimeb" | |
| 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" | |

以字符串形式发布/预订命令标记值

| 表 276: 常量值 | | |
|----------------------------|-------------|-------|
| : | 小数值 | 十六进制值 |
| 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" | |

以空白括起来的字符串形式发布/预订命令标记值

| 表 277: 常量值 | | |
|------------------------------|---------------|-------|
| : | 小数值 | 十六进制值 |
| MQPS_DELETE_PUBLICATION_B | "bDeletePubb" | |
| MQPS_DEREGISTER_PUBLISHER_B | "bDeregPubb" | |
| MQPS_DEREGISTER_SUBSCRIBER_B | "bDeregSubb" | |

| 表 277: 常量值 (继续) | | |
|------------------------------|---------------|-------|
| : | 小数值 | 十六进制值 |
| MQPS_PUBLIC ISH_B | "bPublishb" | |
| MQPS_REGISTER_PUBLIC ISHER_B | "bRegPubb" | |
| MQPS_REGISTER_SUBSCRIBER_B | "bRegSubb" | |
| MQPS_REQUEST_UPDATE_B | "bReqUpdateb" | |

以字符串形式发布/预订选项标记值

| 表 278: 常量值 | | |
|---------------------------------|------------------|-------|
| : | 小数值 | 十六进制值 |
| 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_保留 | "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_改动 | "NoAlter" | |
| MQPS_NO_REG 政务司 | "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_PUBLIC ISH_ON_REQUEST_ONLY | "PubOnReqOnly" | |
| MQPS_RETAIN_PUBLICATION | "RetainPub" | |
| MQPS_VARIABLE_USER_ID | "VariableUserId" | |

以空白括起的字符串形式发布/预订选项标记值

| 表 279: 常量值 | | |
|--------------------------------|--------------------|-------|
| : | 小数值 | 十六进制值 |
| MQPS_ADD_NAME_B | "bAddNameb" | |
| MQPS_ANONYMOUSmous_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_REGLE_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_* (发布/预订选项标记发布/预订命令文件夹 (psc) 标记)

| 表 280: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPSC_FOLDER_VERSION | 1 | X'00000001' |

MQPSC_* (发布/预订选项标记名称)

| 表 281: 常量值 | | |
|--------------------|-----------|-------|
| : | 小数值 | 十六进制值 |
| MQPSC_COMMAND | "Command" | |
| MQPSC_REGLE_OPTION | "RegOpt" | |

| 表 281: 常量值 (继续) | | |
|------------------------------|---------------|-------|
| : | 小数值 | 十六进制值 |
| MQPSC_PUBLICICATION_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_* (发布/预订选项标记 XML 标记名称)

| 表 282: 常量值 | | |
|------------------------------|---------------|-------|
| : | 小数值 | 十六进制值 |
| MQPSC_COMMAND_B | "<Command>" | |
| MQPSC_COMMAND_E | "</Command>" | |
| MQPSC_REGLE_OPTION_B | "<RegOpt>" | |
| MQPSC_REGLE_OPTION_E | "</RegOpt>" | |
| MQPSC_PUBLICICATION_OPTION_B | "<PubOpt>" | |
| MQPSC_PUBLICICATION_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>" | |
| MQPSC_PUBLISH_TIMESTAMP_E | "</PubTime>" | |
| MQPSC_SEQUENCE_NUMBER_B | "<SeqNum>" | |
| MQPSC_SEQUENCE_NUMBER_E | "</SeqNum>" | |

| 表 282: 常量值 (继续) | | |
|--------------------------------|------------------|-------|
| : | 小数值 | 十六进制值 |
| 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_* (以字符串形式发布/预订选项标记值)

| 表 283: 常量值 | | |
|-----------------------------|-------------|-------|
| : | 小数值 | 十六进制值 |
| MQPSC_DELETE_PUBLICATION | "DeletePub" | |
| MQPSC_DEREGISTER_SUBSCRIBER | "DeregSub" | |
| MQPSC_PUBLISH | "Publish" | |
| MQPSC_REGISTER_SUBSCRIBER | "RegSub" | |
| MQPSC_REQUEST_UPDATE | "ReqUpdate" | |

MQPSC_* (以字符串形式发布/预订选项标记值)

| 表 284: 常量值 | | |
|-----------------------------|-----------------|-------|
| : | 小数值 | 十六进制值 |
| 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_留存 | "InformIfRet" | |
| MQPSC_IS_RETAINED_PUB | "IsRetainedPub" | |
| MQPSC_JOIN_SHARED | "JoinShared" | |
| MQPSC_JOIN_EXCLUSIVE | "JoinExcl" | |
| MQPSC_LEAVE_ONLY | "LeaveOnly" | |
| MQPSC_LOCAL | "Local" | |
| 已锁定 MQPSC_LOCKED | "Locked" | |
| MQPSC_NEW_PUBS_ONLY | "NewPubsOnly" | |
| MQPSC_NO_改动 | "NoAlter" | |
| MQPSC_NON_PERSISTENT | "NonPers" | |
| MQPSC_OTHER_SUBS_ONLY | "OtherSubsOnly" | |
| MQPSC_PERSISTENT | "Pers" | |
| MQPSC_PERSISTENT_AS_PUBLISH | "PersAsPub" | |
| MQPSC_PERSISTENT_AS_Q | "PersAsQueue" | |

| 表 284: 常量值 (继续) | | |
|---------------------------|------------------|-------|
| : | 小数值 | 十六进制值 |
| MQPSC_NONE | "None" | |
| MQPSC_PUB_ON_REQUEST_ONLY | "PubOnReqOnly" | |
| MQPSC_RETAIN_PUB | "RetainPub" | |
| MQPSC_VARIABLE_USER_ID | "VariableUserId" | |

MQPSCR_* (发布/预订选项)

发布/预订选项标记发布/预订响应文件夹 (pscr) 标记

| 表 285: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPSCR_FOLDER_VERSION | 1 | X'00000001' |

发布/预订选项标记名称

| 表 286: 常量值 | | |
|-----------------|--------------|-------|
| : | 小数值 | 十六进制值 |
| MQPSCR_完成 | "Completion" | |
| MQPSCR_RESPONSE | "Response" | |
| MQPSCR_REASON | "Reason" | |

发布/预订选项标记 XML 标记名称

| 表 287: 常量值 | | |
|---------------------|-----------------|-------|
| : | 小数值 | 十六进制值 |
| MQPSCR_COMPLETION_B | "<Completion>" | |
| MQPSCR_COMPLETION_E | "</Completion>" | |
| MQPSCR_RESPONSE_B | "<Response>" | |
| MQPSCR_RESPONSE_E | "</Response>" | |
| MQPSCR_REASON_B | "<Reason>" | |
| MQPSCR_REASON_E | "</Reason>" | |

发布/预订选项标记标记值

| 表 288: 常量值 | | |
|----------------|-----------|-------|
| : | 小数值 | 十六进制值 |
| MQPSCR_OK | "ok" | |
| MQPSCR_WARNING | "warning" | |
| MQPSCR_ERROR | "error" | |

MQPSM_* (发布/预订方式)

| 表 289: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPSM_DISABLED | 0 | X'00000000' |

| 表 289: 常量值 (继续) | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPSM_COMPAT | 1 | X'00000001' |
| MQPSM_ENABLED | 2 | X'00000002' |

MQPSPROP_* (发布/预订消息属性)

| 表 290: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPSPROP 无 | 0 | X'00000000' |
| MQPSPROP_COMPAT | 1 | X'00000001' |
| MQPSPROP_RFH2 | 2 | X'00000002' |
| MQPSPROP_MSGPROP | 3 | X'00000003' |

MQPSST_* (命令格式发布/预订状态类型)

| 表 291: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPSST_ALL | 0 | X'00000000' |
| MQPSST_LOCAL | 1 | X'00000001' |
| MQPSST_PARENT | 2 | X'00000002' |
| MQPSST_CHILD | 3 | X'00000003' |

MQPUBO_* (发布/预订发布选项)

| 表 292: 常量值 | | |
|--------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_REG 政务司 | 8 | X'00000008' |
| MQPUBO_IS_RETAINED_PUBLICATION | 16 | X'00000010' |

MQPXP_* (发布/预订路由出口参数结构)

| 表 293: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQPXP_STRUC_ID | "PXP↵" |
| MQPXP_STRUC_ID_ARRAY | 'P', 'X', 'P', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 294: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQPXP_VERSION_1 | 1 | X'00000001' |
| MQPXP_CURRENT_VERSION | 1 | X'00000001' |

MQQA_* (队列属性)

禁止获取值

| 表 295: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQA_GET_INHIBITED | 1 | X'00000001' |
| MQQA_GET_ALLOWED | 0 | X'00000000' |

禁止放置值

| 表 296: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQA_PUT_INHIBITED | 1 | X'00000001' |
| MQQA_PUT_ALLOWED | 0 | X'00000000' |

队列可共享性

| 表 297: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQA_SHAREABLE | 1 | X'00000001' |
| MQQA_NOT_SHAREABLE | 0 | X'00000000' |

回退硬化

| 表 298: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQA_BACKOUT_HARDENED | 1 | X'00000001' |
| MQQA_BACKOUT_NOT_HARDENED | 0 | X'00000000' |

MQQDT_* (队列定义类型)

| 表 299: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQDT_PREDEFINED | 1 | X'00000001' |
| MQQDT_PERMANENT_DYNAMIC | 2 | X'00000002' |
| MQQDT_TEMPORARY_DYNAMIC | 3 | X'00000003' |
| MQQDT_SHARED_DYNAMIC | 4 | X'00000004' |

MQQF_* (队列标志)

| 表 300: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQF_LOCAL_Q | 1 | X'00000001' |
| MQQF_CLWL_USEQ_ANY | 64 | X'00000040' |
| MQQF_CLWL_USEQ_LOCAL | 128 | X'00000080' |

MQQMDT_* (命令格式队列管理器定义类型)

| 表 301: 常量值 | | |
|--------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (队列管理器标志)

| 表 302: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

MQQMFACT_* (命令格式队列管理器设施)

| 表 303: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQMFACT_IMS_BRIDGE | 1 | X'00000001' |
| MQQMFACT_DB2 | 2 | X'00000002' |

MQQMSTA_* (命令格式队列管理器状态)

| 表 304: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQMSTA_START | 1 | X'00000001' |
| MQQMSTA_RUNNING | 2 | X'00000002' |
| MQQMSTA_QUIESCING | 3 | X'00000003' |

MQQMT_* (命令格式队列管理器类型)

| 表 305: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQMT_NORMAL | 0 | X'00000000' |
| MQQMT_REPOSITORY | 1 | X'00000001' |

MQQO_* (命令格式停顿选项)

| 表 306: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQO_YES | 1 | X'00000001' |
| MQQO_NO | 0 | X'00000000' |

MQQSGD_* (队列共享组处置)

| 表 307: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式 QSG 状态)

| 表 308: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式队列服务-时间间隔事件)

| 表 309: 常量值 | | |
|-------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQSIE_NONE | 0 | X'00000000' |
| MQQSIE_HIGH | 1 | X'00000001' |
| MQQSIE_OK | 2 | X'00000002' |

MQQSO_* (SET, BROWSE 和 INPUT 的命令格式队列状态打开选项)

| 表 310: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQSO_NO | 0 | X'00000000' |
| MQQSO_YES | 1 | X'00000001' |
| MQQSO_SHARED | 1 | X'00000001' |
| MQQSO_EXCLUSIVE | 2 | X'00000002' |

MQQSOT_* (命令格式队列状态打开类型)

| 表 311: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQSOT_ALL | 1 | X'00000001' |
| MQQSOT_INPUT | 2 | X'00000002' |

| 表 311: 常量值 (继续) | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQSOT_OUTPUT | 3 | X'00000003' |

MQQSUM_* (命令格式队列状态未落实消息)

| 表 312: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQQSUM_YES | 1 | X'00000001' |
| MQQSUM_NO | 0 | X'00000000' |

MQQT_* (队列类型和扩展队列类型)

队列类型

| 表 313: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

扩展队列类型

| 表 314: 常量值 | | |
|------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQQT_ALL | 1001 | X'000003E9' |

MQRC_* (原因码)

| 表 315: 常量值 | | |
|------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |
| 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' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_禁止 | 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_已达到 | 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' |
| 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_禁止 | 2051 | X'00000803' |
| MQRC_Q_DELETED | 2052 | X'00000804' |

| 表 315: 常量值 (继续) | | |
|------------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |
| 已超过 MQRC_SELECTOR_LIMIT_AUTHORIZED | 2066 | X'00000812' |
| MQRC_SELECTOR_ERROR | 2067 | X'00000813' |
| MQRC_SELECTOR_NOT_FOR_TYPE | 2068 | X'00000814' |
| MQRC_SIGNAL_众数 | 2069 | X'00000815' |
| MQRC_SIGNAL_REQUEST_接受 | 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_RECEIVED | 2079 | X'0000081F' |
| MQRC_TRUNCATED_MSG_FAILED | 2080 | X'00000820' |
| MQRC_UNKNOWN_ALIAS_BASE_Q | 2082 | X'00000822' |
| MQRC_UNKNOWN_OBJECT_NAME | 2085 | X'00000825' |
| MQRC_UNKNOWN_OBJECT_Q_MGR | 2086 | X'00000826' |
| MQRC_UNKNOWN_REMOTE_Q_MGR | 2087 | X'00000827' |
| MQRC_WAIT_INTERVAL_ERROR | 2090 | X'0000082A' |
| MQRC_XMIT_Q_TYPE_ERROR | 2091 | X'0000082B' |
| MQRC_XMIT_Q_USAGE_ERROR | 2092 | X'0000082C' |
| MQRC_NOT_OPEN_FOR_PASS_ALL | 2093 | X'0000082D' |
| MQRC_NOT_OPEN_FOR_PASS_制 | 2094 | X'0000082E' |
| MQRC_NOT_OPEN_FOR_SET_ALL | 2095 | X'0000082F' |
| MQRC_NOT_OPEN_FOR_SET_制 | 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' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_CANCEL | 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_汇率 | 2119 | X'00000847' |
| MQRC_CONVERTED_MSG_TOO_BIG | 2120 | X'00000848' |
| MQRC_截断 | 2120 | X'00000848' |
| MQRC_NO_EXTERNAL_参与者 | 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' |
| 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_原因 | 2136 | X'00000858' |
| MQRC_OPEN_FAILED | 2137 | X'00000859' |
| MQRC_ADAPTER_DISC_LOAD_ERROR | 2138 | X'0000085A' |
| MQRC_CNO_ERROR | 2139 | X'0000085B' |

| 表 315: 常量值 (继续) | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_限制 | 2187 | X'0000088B' |
| MQRC_STOPPED_BY_CLUSTER_EXIT | 2188 | X'0000088C' |
| MQRC_CLUSTER_RESOLUTION_ERROR | 2189 | X'0000088D' |
| MQRC_CONVERTED_STRING_TOO_BIG | 2190 | X'0000088E' |
| 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' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_审定 | 2216 | X'000008A8' |
| MQRC_CONNECTION_NOT_AUTHORIZED | 2217 | X'000008A9' |
| MQRC_MSG_TOO_BIG_FOR_CHANNEL | 2218 | X'000008AA' |
| MQRC_CALL_IN_PROGRESS | 2219 | X'000008AB' |
| MQRC_RMH_ERROR | 2220 | X'000008AC' |
| MQRC_Q_MGR_ACTIVE | 2222 | X'000008AE' |
| MQRC_Q_MGR_NOT_ACTIVE | 2223 | X'000008AF' |
| MQRC_Q_DEPTH_HIGH | 2224 | X'000008B0' |
| MQRC_Q_DEPTH_LOW | 2225 | X'000008B1' |
| MQRC_Q_SERVICE_INTERVAL_HIGH | 2226 | X'000008B2' |
| MQRC_Q_SERVICE_INTERVAL_OK | 2227 | X'000008B3' |
| MQRC_RFH_HEADER_FIELD_ERROR | 2228 | X'000008B4' |
| MQRC_RAS_PROPERTY_ERROR | 2229 | X'000008B5' |
| MQRC_UNIT_OF_WORK_NOT_STARTED | 2232 | X'000008B8' |
| MQRC_CHANNEL_AUTO_DEF_OK | 2233 | X'000008B9' |
| MQRC_CHANNEL_AUTO_DEF_ERROR | 2234 | X'000008BA' |
| MQRC_CFH_ERROR | 2235 | X'000008BB' |
| MQRC_CFIL_ERROR | 2236 | X'000008BC' |
| MQRC_CFIN_ERROR | 2237 | X'000008BD' |
| MQRC_CFSL_ERROR | 2238 | X'000008BE' |
| MQRC_CFST_ERROR | 2239 | X'000008BF' |
| MQRC_INCOMPLETE_GROUP | 2241 | X'000008C1' |
| MQRC_INCOMPLETE_MSG | 2242 | X'000008C2' |
| MQRC_INCONSISTENT_CCSDS | 2243 | X'000008C3' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_不法_gmo_version | 2256 | X'000008D0' |
| MQRC_不法_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_禁止 | 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_汇率 | 2272 | X'000008E0' |
| MQRC_CONNECTION_ERROR | 2273 | X'000008E1' |
| MQRC_OPTION_ENVIRONMENT_ERROR | 2274 | X'000008E2' |
| MQRC_CD_ERROR | 2277 | X'000008E5' |
| MQRC_CLIENT_CONN_ERROR | 2278 | X'000008E6' |
| MQRC_CHANNEL_STOPPED_BY_USER | 2279 | X'000008E7' |
| MQRC_HCONFIG_ERROR | 2280 | X'000008E8' |
| MQRC_FUNCTION_ERROR | 2281 | X'000008E9' |
| MQRC_CHANNEL_STARTED | 2282 | X'000008EA' |
| MQRC_CHANNEL_STOPPED | 2283 | X'000008EB' |
| MQRC_CHANNEL_CONV_ERROR | 2284 | X'000008EC' |
| MQRC_SERVICE_NOT_AVAILABLE | 2285 | X'000008ED' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_截断 | 2311 | X'00000907' |
| MQRC_SELECTOR_WRONG_TYPE | 2312 | X'00000908' |
| MQRC_INCONSISTENT_ITEM_TYPE | 2313 | X'00000909' |
| MQRC_INDEX_ERROR | 2314 | X'0000090A' |
| MQRC_SYSTEM_BAG_NOT_ALTERABLE | 2315 | X'0000090B' |
| MQRC_ITEM_COUNT_ERROR | 2316 | X'0000090C' |
| MQRC_FORMAT_NOT_SUPPORTED | 2317 | X'0000090D' |
| MQRC_SELECTOR_NOT_SUPPORTED | 2318 | X'0000090E' |
| MQRC_ITEM_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_QUIRY_COMMAND_ERROR | 2324 | X'00000914' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQRC_NESTED_BAG_NOT_SUPPORTED | 2325 | X'00000915' |
| MQRC_BAG_不法类型 | 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' |
| 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_不兼容 | 2360 | X'00000938' |
| MQRC_NEXT_RECORD_ERROR | 2361 | X'00000939' |
| MQRC_BACKOUT_THRESHOLD_已达到 | 2362 | X'0000093A' |
| MQRC_MSG_NOT_FOUND | 2363 | X'0000093B' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQRC_JMS_FORMAT_ERROR | 2364 | X'0000093C' |
| MQRC_SEGMENTS_NOT_SUPPORTED | 2365 | X'0000093D' |
| MQRC_不法_cf_level | 2366 | X'0000093E' |
| MQRC_CONFIG_CREATE_OBJECT | 2367 | X'0000093F' |
| MQRC_CONFIG_CHANGE_OBJECT | 2368 | X'00000940' |
| MQRC_CONFIG_DELETE_OBJECT | 2369 | X'00000941' |
| MQRC_CONFIG_REFRESH_OBJECT | 2370 | X'00000942' |
| MQRC_CHANNEL_SSL_ERROR | 2371 | X'00000943' |
| MQRC_PARTICIPANT_NOT_DEFINED | 2372 | X'00000944' |
| MQRC_CF_STRUC_FAILED | 2373 | X'00000945' |
| MQRC_API_EXIT_ERROR | 2374 | X'00000946' |
| MQRC_API_EXIT_INIT_ERROR | 2375 | X'00000947' |
| MQRC_API_EXIT_TERM_ERROR | 2376 | X'00000948' |
| MQRC_EXIT_REASON_ERROR | 2377 | X'00000949' |
| MQRC_RESERVED_VALUE_ERROR | 2378 | X'0000094A' |
| MQRC_NO_DATA_AVAILABLE | 2379 | X'0000094B' |
| MQRC_SCO_ERROR | 2380 | X'0000094C' |
| MQRC_KEY_REPOSITORY_ERROR | 2381 | X'0000094D' |
| MQRC_CRYPTO_HARDWARE_ERROR | 2382 | X'0000094E' |
| MQRC_AUTH_INFO_REC_COUNT_ERROR | 2383 | X'0000094F' |
| MQRC_AUTH_INFO_REC_ERROR | 2384 | X'00000950' |
| MQRC_AIR_ERROR | 2385 | X'00000951' |
| MQRC_AUTH_INFO_TYPE_ERROR | 2386 | X'00000952' |
| MQRC_AUTH_INFO_CONN_NAME_ERROR | 2387 | X'00000953' |
| MQRC_LDAP_USER_NAME_ERROR | 2388 | X'00000954' |
| MQRC_LDAP_USER_NAME_LENGTH_ERR | 2389 | X'00000955' |
| MQRC_LDAP_PASSWORD_ERROR | 2390 | X'00000956' |
| MQRC_SSL_ALREADY_INITIALIZED | 2391 | X'00000957' |
| MQRC_SSL_CONFIG_ERROR | 2392 | X'00000958' |
| MQRC_SSL_INITIALIZATION_ERROR | 2393 | X'00000959' |
| MQRC_Q_INDEX_TYPE_ERROR | 2394 | X'0000095A' |
| MQRC_CFBS_ERROR | 2395 | X'0000095B' |
| MQRC_SSL_NOT_ALLOWED | 2396 | X'0000095C' |
| MQRC_JSSE_ERROR | 2397 | X'0000095D' |
| MQRC_SSL_PEER_NAME_MISMATCH | 2398 | X'0000095E' |
| MQRC_SSL_PEER_NAME_ERROR | 2399 | X'0000095F' |
| MQRC_UNSUPPORTED_CIPHER_SUITE | 2400 | X'00000960' |
| MQRC_SSL_CERTIFICATE_REVOKED | 2401 | X'00000961' |
| MQRC_SSL_CERT_STORE_ERROR | 2402 | X'00000962' |

| 表 315: 常量值 (继续) | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_MATCH | 2434 | X'00000982' |
| MQRC_ALTER_SUB_ERROR | 2435 | X'00000983' |
| MQRC_DURABILITY_NOT_ALLOWED | 2436 | X'00000984' |
| MQRC_NO_RETAINED_MSG | 2437 | X'00000985' |
| MQRC_SRO_ERROR | 2438 | X'00000986' |
| MQRC_SUB_NAME_ERROR | 2440 | X'00000988' |
| MQRC_OBJECT_STRING_ERROR | 2441 | X'00000989' |
| MQRC_PROPERTY_NAME_ERROR | 2442 | X'0000098A' |
| MQRC_SEGMENTATION_NOT_ALLOWED | 2443 | X'0000098B' |
| MQRC_CBD_ERROR | 2444 | X'0000098C' |
| MQRC_CTLO_ERROR | 2445 | X'0000098D' |
| MQRC_NO_CALLBACKS_ACTIVE | 2446 | X'0000098E' |
| MQRC_CALLBACK_NOT_REGISTERED | 2448 | X'00000990' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_留存 | 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_汇率 | 2492 | X'000009BC' |
| MQRC_GET_ENABLED | 2494 | X'000009BE' |
| MQRC_MODULE_NOT_FOUND | 2495 | X'000009BF' |
| MQRC_MODULE_INVALID | 2496 | X'000009C0' |
| MQRC_MODULE_ENTRY_NOT_FOUND | 2497 | X'000009C1' |
| MQRC_MIXED_CONTENT_NOT_ALLOWED | 2498 | X'000009C2' |
| MQRC_MSG_HANDLE_IN_USE | 2499 | X'000009C3' |
| MQRC_HCONN_ASYNC_ACTIVE | 2500 | X'000009C4' |
| MQRC_MHBO_ERROR | 2501 | X'000009C5' |
| MQRC_PUBLICATION_FAILURE | 2502 | X'000009C6' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQRC_SUB_禁止 | 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_交付 | 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_禁止 | 2531 | X'000009E3' |
| MQRC_MSG_HANDLE_COPY_FAILURE | 2532 | X'000009E4' |
| MQRC_DEST_CLASS_NOT_ALTERABLE | 2533 | X'000009E5' |
| MQRC_OPERATION_NOT_ALLOWED | 2534 | X'000009E6' |
| MQRC_ACTION_ERROR | 2535 | X'000009E7' |
| MQRC_CHANNEL_NOT_AVAILABLE | 2537 | X'000009E9' |
| MQRC_HOST_NOT_AVAILABLE | 2538 | X'000009EA' |
| MQRC_CHANNEL_CONFIG_ERROR | 2539 | X'000009EB' |
| MQRC_UNKNOWN_CHANNEL_NAME | 2540 | X'000009EC' |
| MQRC_LOOPING_PUBLICATION | 2541 | X'000009ED' |
| MQRC_ALREADY_联接 | 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' |

| 表 315: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_IN 区内的缓冲区 | 6113 | X'000017E1' |
| MQRC_IN 有数据 | 6114 | X'000017E2' |
| MQRC_DATA_截断 | 6115 | X'000017E3' |
| MQRC_ZERO_LENGTH | 6116 | X'000017E4' |
| MQRC_NEGTIVE_LENGTH | 6117 | X'000017E5' |
| MQRC_NEGTIVE_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_不法版本 | 6128 | X'000017F0' |
| MQRC_REFERENCE_ERROR | 6129 | X'000017F1' |

MQRCCF_* (命令格式头原因码)

| 表 316: 常量值 | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 316: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |
| 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' |

| 表 316: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_CCSSID_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' |
| MQRCCF_REG_OPTIONS_ERROR | 3083 | X'00000C0B' |
| MQRCCF_PUB_OPTIONS_ERROR | 3084 | X'00000C0C' |
| MQRCCF_UNKNOWN_BROKER | 3085 | X'00000C0D' |
| MQRCCF_Q_MGR_CCSSID_ERROR | 3086 | X'00000C0E' |
| MQRCCF_DEL_OPTIONS_ERROR | 3087 | X'00000C0F' |
| MQRCCF_CLUSTER_NAME_CONFLICT | 3088 | X'00000C10' |

| 表 316: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |
| 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' |

| 表 316: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| MQRCCF_PARM_CONFLICT | 3203 | X'00000C83' |
| MQRCCF_COMMAND_禁止 | 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' |
| 已超过 MQRCCF_OBJECT_LIMIT_额 | 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' |
| 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' |

| 表 316: 常量值 (继续) | | |
|---------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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' |
| 找到 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'00000CC5' |
| 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' |
| 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_禁止 | 3318 | X'00000CF6' |

| 表 316: 常量值 (继续) | | |
|-------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_不法_chlauth_action | 3339 | X'00000D0B' |
| MQRCCF_WRONG_CHLAUTH_USERSRC | 3340 | X'00000D0C' |
| MQRCCF_CHLAUTH_WARN_ERROR | 3341 | X'00000D0D' |
| MQRCCF_差错_chlauth_match | 3342 | X'00000D0E' |
| MQRCCF_IPADDR_RANGE_CONFLICT | 3343 | X'00000D0F' |
| 已超过 MQRCCF_CHLAUTH_MAX_团 | 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' |
| 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' |

| 表 316: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_正在终止 | 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_不法类型 | 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' |
| 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' |

| 表 316: 常量值 (继续) | | |
|--------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_不法类型 | 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_* (客户机重新连接常量)

| 表 317: 常量值 | | |
|-------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRCN_NO | 0 | X'00000000' |
| MQRCN_YES | 1 | X'00000001' |
| MQRCN_Q_MGR | 2 | X'00000002' |

| 表 317: 常量值 (继续) | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRCN_DISABLED | 3 | X'00000003' |

MQRCVTIME_* (接收超时类型)

| 表 318: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRCVTIME_MULTIPLY | 0 | X'00000000' |
| MQRCVTIME_ADD | 1 | X'00000001' |
| MQRCVTIME_EQUAL | 2 | X'00000002' |

MQREADA_* (预读值)

| 表 319: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQREADA_NO | 0 | X'00000000' |
| MQREADA_YES | 1 | X'00000001' |
| MQREADA_DISABLED | 2 | X'00000002' |
| 已禁止 MQREADA_ALLOWED | 3 | X'00000003' |
| MQREADA_待办事宜 | 4 | X'00000004' |

MQRECORDING_* (记录选项)

| 表 320: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRECORDING_DISABLED | 0 | X'00000000' |
| MQRECORDING_Q | 1 | X'00000001' |
| MQRECORDING_MSG | 2 | X'00000002' |

MQREGO_* (发布/预订注册选项)

| 表 321: 常量值 | | |
|----------------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_PUBLICISH_ON_REQUEST_ONLY | 32 | X'00000020' |
| MQREGO_DEREGISTER_ALL | 64 | X'00000040' |
| MQREGO_INCLUDE_STREAM_NAME | 128 | X'00000080' |
| MQREGO_INFORM_IF_留存 | 256 | X'00000100' |
| MQREGO_DUPLICATES_OK | 512 | X'00000200' |
| MQREGO_NON_PERSISTENT | 1024 | X'00000400' |

| 表 321: 常量值 (继续) | | |
|------------------------------|---------|-------------|
| : | 小数值 | 十六进制值 |
| 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_改动 | 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_* (规则和格式化头结构以及标志)

规则和格式化头结构

| 表 322: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQRFH_STRUC_ID | "RFH↵" |
| MQRFH_STRUC_ID_ARRAY | 'R', 'F', 'H', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 323: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

规则和格式化头标志

| 表 324: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRFH_NONE | 0 | X'00000000' |
| MQRFH_NO_FLAGS | 0 | X'00000000' |

MQRFH2_* (发布/预订选项标记 RFH2 顶级文件夹标记)

| 表 325: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRFH2_NAME_VALUE_VERSION | 1 | X'00000001' |

MQRFH2_* (发布/预订选项标记名称)

| 表 326: 常量值 | | |
|---------------------------|--------|-------|
| : | 小数值 | 十六进制值 |
| MQRFH2_PUBSUB_CMD_FOLDER | "psc" | |
| MQRFH2_PUBSUB_RESP_FOLDER | "pscr" | |
| MQRFH2_MSG_CONTENT_FOLDER | "mcd" | |
| MQRFH2_USER_FOLDER | "usr" | |

MQRFH2_* (发布/预订选项标记 XML 标记名称)

| 表 327: 常量值 | | |
|-----------------------------|-----------|-------|
| : | 小数值 | 十六进制值 |
| 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_* (返回长度)

| 表 328: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRL_UNDEFINED | -1 | X'FFFFFFFF' |

MQRMH_* (参考消息头结构)

| 表 329: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQRMH_STRUC_ID | "RMH↵" |
| MQRMH_STRUC_ID_ARRAY | 'R', 'M', 'H', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 330: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRMH_VERSION_1 | 1 | X'00000001' |
| MQRMH_CURRENT_VERSION | 1 | X'00000001' |

MQRMHF_* (参考消息头标志)

| 表 331: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRMHF_LAST | 1 | X'00000001' |
| MQRMHF_NOT_LAST | 0 | X'00000000' |

MQRO_* (报告选项)

| 表 332: 常量值 | | |
|--------------------------------|-----------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (报告选项掩码)

| 表 333: 常量值 | | |
|--------------------------------|------------|-------------|
| : | 小数值 | 十六进制值 |
| MQRO_REJECT_UNSUP_MASK | 270270464 | X'101C0000' |
| MQRO_ACCEPT_UNSUP_MASK | -270532353 | X'EFE000FF' |
| MQRO_ACCEPT_UNSUP_IF_XMIT_MASK | 261888 | X'0003FF00' |

MQROUTE_* (跟踪路径)

跟踪路由最大活动数 (MQIACF_MAX_ACTIVactivities)

| 表 334: 常量值 | | |
|------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQROUTE_UNLIMITED_ACTIVITIES | 0 | X'00000000' |

跟踪路由详细信息 (MQIACF_ROUTE_DETAIL)

| 表 335: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQROUTE_DETAIL_LOW | 2 | X'00000002' |
| MQROUTE_DETAIL_MEDIUM | 8 | X'00000008' |
| MQROUTE_DETAIL_HIGH | 32 | X'00000020' |

跟踪路由转发 (MQIACF_ROUTE_转发)

| 表 336: 常量值 | | |
|--------------------------------|--------|-------------|
| : | 小数值 | 十六进制值 |
| MQROUTE_FORWARD_ALL | 256 | X'00000100' |
| MQROUTE_FORWARD_IF_SUPPORTED | 512 | X'00000200' |
| MQROUTE_FORWARD_REJ_UNSUP_MASK | -65536 | X'FFFF0000' |

跟踪路径传递 (MQIACF_ROUTE_DELIVERY)

| 表 337: 常量值 | | |
|--------------------------------|--------|-------------|
| : | 小数值 | 十六进制值 |
| MQROUTE_DELIVER_YES | 4096 | X'00001000' |
| MQROUTE_DELIVER_NO | 8192 | X'00002000' |
| MQROUTE_DELIVER_REJ_UNSUP_MASK | -65536 | X'FFFF0000' |

跟踪路由累积 (MQIACF_ROUTE_累加)

| 表 338: 常量值 | | |
|-----------------------|-------|-------------|
| : | 小数值 | 十六进制值 |
| MQROUTE_累计无 | 65539 | X'00010003' |
| MQROUTE_累加_in_msg | 65540 | X'00010004' |
| MQROUTE_蓄电池_and_reply | 65541 | X'00010005' |

MQRP_* (命令格式替换选项)

| 表 339: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRP_YES | 1 | X'00000001' |
| MQRP_NO | 0 | X'00000000' |

MQRQ_* (命令格式原因限定符)

| 表 340: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 340: 常量值 (继续) | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRQ_Q_MGR_正在停止 | 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_撤销 | 19 | X'00000013' |

MQRT_* (命令格式刷新类型)

| 表 341: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRT_CONFIGURATION | 1 | X'00000001' |
| MQRT_到期 | 2 | X'00000002' |
| MQRT_NSPROC | 3 | X'00000003' |
| MQRT_PROXYSUB | 4 | X'00000004' |

MQRU_* (仅请求)

| 表 342: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQRU_PUBLIC ISH_ON_REQUEST | 1 | X'00000001' |
| MQRU_PUBLISH_ALL | 2 | X'00000002' |

MQSCA_* (SSL 客户机认证)

| 表 343: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSCA_REQUIRED | 0 | X'00000000' |
| MQSCA_OPTIONAL | 1 | X'00000001' |

MQSCO_* (SSL 配置选项)

SSL 配置选项结构

| 表 344: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQSCO_STRUC_ID | "SCO" |
| MQSCO_STRUC_ID_ARRAY | 'S','C','O',' ' |

注: 符号 - 表示单个空白字符。

| 表 345: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

注: 符号 - 表示单个空白字符。

SSL 配置选项密钥重置计数

| 表 346: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSCO_RESET_COUNT_DEFAULT | 0 | X'00000000' |

命令格式队列定义作用域

| 表 347: 常量值 | | |
|-------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSCO_Q_MGR | 1 | X'00000001' |
| MQSCO_CELL | 2 | X'00000002' |

MQSCOPE_* (发布作用域)

| 表 348: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSCOPE_ALL | 0 | X'00000000' |
| MQSCOPE_AS_PARENT | 1 | X'00000001' |
| MQSCOPE_QMGR | 4 | X'00000004' |

MQSCYC_* (安全案例)

| 表 349: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSCYC_UPPER | 0 | X'00000000' |
| MQSCYC_MIXED | 1 | X'00000001' |

MQSD_* (对象描述符结构)

| 表 350: 常量名称和结构 | |
|---------------------|-----------------|
| : | 结构 |
| MQSD_STRUC_ID | "SD↵" |
| MQSD_STRUC_ID_ARRAY | 'S','D','↵','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 351: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSD_VERSION_1 | 1 | X'00000001' |
| MQSD_CURRENT_VERSION | 1 | X'00000001' |

MQSECITEM_* (命令格式安全性项)

| 表 352: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_MQCMLS | 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_* (命令格式安全交换机和交换机州)

命令格式安全开关

| 表 353: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSECSW_PROCESS | 1 | X'00000001' |
| MQSECSW_NAMELIST | 2 | X'00000002' |
| MQSECSW_Q | 3 | X'00000003' |
| MQSECSW_TOPIC | 4 | X'00000004' |
| MQSECSW_CONTEXT | 6 | X'00000006' |
| MQSECSW_ALTERNATE_USER | 7 | X'00000007' |
| MQSECSW_COMMAND | 8 | X'00000008' |
| MQSECSW_CONNECTION | 9 | X'00000009' |
| MQSECSW_SUBSYSTEM | 10 | X'0000000A' |

| 表 353: 常量值 (继续) | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSECSW_COMMAND_RESOURCES | 11 | X'0000000B' |
| MQSECSW_Q_MGR | 15 | X'0000000F' |
| MQSECSW_QSG | 16 | X'00000010' |

命令格式安全开关状态

| 表 354: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_OVER 覆盖 | 26 | X'0000001A' |

MQSECTYPE_* (命令格式安全性类型)

| 表 355: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSECTYPE_AUTHSERV | 1 | X'00000001' |
| MQSECTYPE_SSL | 2 | X'00000002' |
| MQSECTYPE_CLASSES | 3 | X'00000003' |

MQSEG_* (分段)

| 表 356: 常量名称和值 | |
|---------------|-----|
| : | 值 |
| MQSEG_禁止 | '-' |
| MQSEG_ALLOWED | 'A' |

注: 符号 - 表示单个空白字符。

MQSEL_* (特殊选择器值)

| 表 357: 常量值 | | |
|----------------------------|--------|-------------|
| : | 小数值 | 十六进制值 |
| MQSEL_ANY_SELECTOR | -30001 | X'FFFF8ACF' |
| MQSEL_ANY_USER_SELECTOR | -30002 | X'FFFF8ACE' |
| MQSEL_ANY_SYSTEM_SELECTOR | -30003 | X'FFFF8ACD' |
| MQSEL_ALL_SELECTORS | -30001 | X'FFFF8ACF' |
| MQSEL_ALL_USER_SELECTORS | -30002 | X'FFFF8ACE' |
| MQSEL_ALL_SYSTEM_SELECTORS | -30003 | X'FFFF8ACD' |

MQSELTYPE_* (选择器类型)

| 表 358: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSELTYPE_NONE | 0 | X'00000000' |
| MQSELTYPE_STANDARD | 1 | X'00000001' |
| MQSELTYPE_EXTENDED | 2 | X'00000002' |

MQSID_* (安全标识)

| 表 359: 常量名称和值 | |
|------------------|--------------------------|
| : | 值 |
| MQSID_NONE | X'00...00' (40 个空值) |
| MQSID_NONE_ARRAY | '\0', '\0', ... (40 个空值) |

MQSIDT_* (安全标识类型)

| 表 360: 常量名称和值 | |
|------------------------|-------|
| : | 十六进制值 |
| MQSIDT_NONE | X'00' |
| MQSIDT_NT_SECURITY_ID | X'01' |
| MQSIDT_WAS_SECURITY_ID | X'02' |

MQSMPO_* (设置消息属性选项和结构)

设置消息属性选项结构

| 表 361: 常量结构 | |
|-----------------------|--------------------|
| : | 结构 |
| MQSMPO_STRUC_ID | "SMPO" |
| MQSMPO_STRUC_ID_ARRAY | 'S', 'M', 'P', 'O' |

注: 符号 - 表示单个空白字符。

| 表 362: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSMPO_VERSION_1 | 1 | X'00000001' |
| MQSMPO_CURRENT_VERSION | 1 | X'00000001' |

设置消息属性选项

| 表 363: 常量值 | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 363: 常量值 (继续) | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSMPO_NONE | 0 | X'00000000' |

MQSO_* (预订选项)

| 表 364: 常量值 | | |
|--------------------------------|-----------|-------------|
| : | 小数值 | 十六进制值 |
| MQSO_NONE | 0 | X'00000000' |
| MQSO_non_持久 | 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_持久 | 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_PUBLICICATIONS_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_* (同步点可用性)

| 表 365: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSP_AVAILABLE | 1 | X'00000001' |
| MQSP_NOT_AVAILABLE | 0 | X'00000000' |

MQSQQM_* (共享队列管理器名称)

| 表 366: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSQQM_USE | 0 | X'00000000' |
| MQSQQM_IGNORE | 1 | X'00000001' |

MQSR_* (操作)

| 表 367: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSR_ACTION_PUBLICATION | 1 | X'00000001' |

MQSRO_* (预订请求选项结构)

| 表 368: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQSRO_STRUC_ID | "SR0-" |
| MQSRO_STRUC_ID_ARRAY | 'S','R','0','-' |

注: 符号 - 表示单个空白字符。

| 表 369: 常量值 | | |
|-------------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (段状态)

| 表 370: 常量名称和结构 | |
|--------------------|-----|
| : | 结构 |
| MQSS_NOT_A_SEGMENT | '-' |
| MQSS_SEGMENT | 'S' |
| MQSS_LAST_SEGMENT | 'L' |

注: 符号 - 表示单个空白字符。

MQSSL_* (SSL FIPS 需求)

| 表 371: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSSL_FIPS_NO | 0 | X'00000000' |
| MQSSL_FIPS_YES | 1 | X'00000001' |

MQSTAT_* (统计信息选项)

| | | |
|--------------------------------|---|-------------|
| MQSTAT_TYPE_ASYNC_ERROR | 0 | X'00000000' |
| MQSTAT_TYPE_RECONNECTION | 0 | X'00000000' |
| MQSTAT_TYPE_RECONNECTION_ERROR | 0 | X'00000000' |

MQSTS_* (状态报告结构)

| 表 372: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQSTS_STRUC_ID | "STAT" |
| MQSTS_STRUC_ID_ARRAY | 'S','T','A','T' |

注: 符号 表示单个空白字符。

| 表 373: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSTS_VERSION_1 | 1 | X'00000001' |
| MQSTS_CURRENT_VERSION | 1 | X'00000001' |

MQSUB_* (持久预订)

持久预订

| 表 374: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSUB_DURABLE_AS_PARENT | 0 | X'00000000' |
| MQSUB_DURABLE_ALLOWED | 1 | X'00000001' |
| MQSUB_DURABLE_DISABLED | 2 | X'00000002' |

持久预订

| 表 375: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSUB_DURABLE_ALL | -1 | X'FFFFFFFF' |
| MQSUB_DURABLE_YES | 1 | X'00000001' |
| MQSUB_DURABLE_NO | 2 | X'00000002' |

MQSUBTYPE_* (命令格式预订类型)

| 表 376: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (命令格式 "暂挂状态")

| 表 377: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSUS_YES | 1 | X'00000001' |
| MQSUS_NO | 0 | X'00000000' |

MQSVC_* (服务)

服务类型

| 表 378: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSVC_TYPE_COMMAND | 0 | X'00000000' |
| MQSVC_TYPE_SERVER | 1 | X'00000001' |

服务控制

| 表 379: 常量值 | | |
|---------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSVC_CONTROL_Q_MGR | 0 | X'00000000' |
| MQSVC_CONTROL_Q_MGR_START | 1 | X'00000001' |
| MQSVC_CONTROL_MANUAL | 2 | X'00000002' |

服务状态

| 表 380: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSVC_STATUS_STOPPED | 0 | X'00000000' |
| MQSVC_STATUS_STARTING | 1 | X'00000001' |
| MQSVC_STATUS_RUNNING | 2 | X'00000002' |
| MQSVC_STATUS正在停止 | 3 | X'00000003' |
| MQSVC_STATUS_RESTARTING | 4 | X'00000004' |

MQSYNCPOINT_* (用于发布/预订迁移的命令格式同步点值)

| 表 381: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSYNCPOINT_YES | 0 | X'00000000' |
| MQSYNCPOINT_IFPER | 1 | X'00000001' |

MQSYSP_* (命令格式系统参数值)

| 表 382: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQSYSP_NO | 0 | X'00000000' |
| 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' |

| 表 382: 常量值 (继续) | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (主题属性)

通配符

| 表 383: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTA_BLOCK | 1 | X'00000001' |
| MQTA_PASSTHRU | 2 | X'00000002' |

允许的预订

| 表 384: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTA_SUB_AS_PARENT | 0 | X'00000000' |
| MQTA_SUB_禁止 | 1 | X'00000001' |
| MQTA_SUB_ALLOWED | 2 | X'00000002' |

代理子传播

| 表 385: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTA_PROXY_SUB_FORCE | 1 | X'00000001' |
| MQTA_PROXY_SUB_FIRSTUSE | 2 | X'00000002' |

允许发布

| 表 386: 常量值 | | |
|--------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTA_PUB_AS_PARENT | 0 | X'00000000' |
| MQTA_PUB_禁止 | 1 | X'00000001' |
| MQTA_PUB_ALLOWED | 2 | X'00000002' |

MQTC_* (触发器控制)

| 表 387: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTC_OFF | 0 | X'00000000' |
| MQTC_ON | 1 | X'00000001' |

MQTCPKEEP_* (TCP 保持活动)

| 表 388: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTCPKEEP_NO | 0 | X'00000000' |
| MQTCPKEEP_YES | 1 | X'00000001' |

MQTCPSTACK_* (TCP 堆栈类型)

| 表 389: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTCPSTACK_SINGLE | 0 | X'00000000' |
| MQTCPSTACK_MULTIPLE | 1 | X'00000001' |

MQTIME_* (命令格式时间单位)

| 表 390: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTIME_UNIT_MINS | 0 | X'00000000' |
| MQTIME_UNIT_SECS | 1 | X'00000001' |

MQTM_* (触发器消息结构)

| 表 391: 常量结构 | |
|---------------------|-----------------|
| : | 结构 |
| MQTM_STRUC_ID | "TM↵" |
| MQTM_STRUC_ID_ARRAY | 'T','M','↵','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 392: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTM_VERSION_1 | 1 | X'00000001' |
| MQTM_CURRENT_VERSION | 1 | X'00000001' |

MQTMC_* (触发器消息字符格式结构)

| 表 393: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQTMC_STRUC_ID | "TMC↵" |
| MQTMC_STRUC_ID_ARRAY | 'T','M','C','↵' |
| MQTMC_VERSION_1 | "↵↵1" |

| 表 393: 常量结构 (继续) | |
|-----------------------------|--------------------|
| : | 结构 |
| MQTMC_VERSION_2 | "- - - 2" |
| MQTMC_CURRENT_VERSION | "- - - 2" |
| MQTMC_VERSION_1_ARRAY | '-', '-', '-', '1' |
| MQTMC_VERSION_2_ARRAY | '-', '-', '-', '2' |
| MQTMC_CURRENT_VERSION_ARRAY | '-', '-', '-', '2' |

注: 符号 - 表示单个空白字符。

MQTOPT_* (主题类型)

| 表 394: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTOPT_LOCAL | 0 | X'00000000' |
| MQTOPT_CLUSTER | 1 | X'00000001' |
| MQTOPT_ALL | 2 | X'00000002' |

MQTRAXSTR_* (通道启动程序跟踪自动启动)

| 表 395: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTRAXSTR_NO | 0 | X'00000000' |
| MQTRAXSTR_YES | 1 | X'00000001' |

MQTSCOPE_* (预订作用域)

| 表 396: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTSCOPE_QMGR | 1 | X'00000001' |
| MQTSCOPE_ALL | 2 | X'00000002' |

MQTT_* (触发器类型)

| 表 397: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTT_NONE | 0 | X'00000000' |
| MQTT_FIRST | 1 | X'00000001' |
| MQTT EVERY | 2 | X'00000002' |
| MQTT_DEPTH | 3 | X'00000003' |

MQTYPE_* (属性数据类型)

| 表 398: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQTYPE_AS_SET | 0 | X'00000000' |
| MQTYPE_NULL | 2 | X'00000002' |

| 表 398: 常量值 (继续) | | |
|--------------------|------|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (发布/预订用户属性选择器)

| 表 399: 常量值 | | |
|------------|----------|-------------|
| : | 小数值 | 十六进制值 |
| MQUA_FIRST | 65536 | X'00010000' |
| MQUA_LAST | 99999999 | X'3B9AC9FF' |

MQUIDSUPP_* (命令格式用户标识支持)

| 表 400: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQUIDSUPP_NO | 0 | X'00000000' |
| MQUIDSUPP_YES | 1 | X'00000001' |

MQUNDELIVERED_* (用于发布/预订迁移的命令格式未交付的值)

| 表 401: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQUNDELIVERED_NORMAL | 0 | X'00000000' |
| MQUNDELIVERED_SAFE | 1 | X'00000001' |
| MQUNDELIVERED_DISCARD | 2 | X'00000002' |
| MQUNDELIVERED_KEEP | 3 | X'00000003' |

MQUOWST_* (命令格式 UOW 状态)

| 表 402: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQUOWST_NONE | 0 | X'00000000' |
| MQUOWST_ACTIVE | 1 | X'00000001' |
| MQUOWST_PREPARED | 2 | X'00000002' |
| MQUOWST_未决 | 3 | X'00000003' |

MQUOWT_* (命令格式 UOW 类型)

| 表 403: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (队列使用情况)

| 表 404: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQUS_NORMAL | 0 | X'00000000' |
| MQUS_TRANSMISSION | 1 | X'00000001' |

MQUSAGE_* (命令格式页集使用值和数据集使用值)

命令格式页集使用情况值

| 表 405: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

命令格式数据集使用情况值

| 表 406: 常量值 | | |
|-------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (值长度)

| 表 407: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQVL_NULL_TERMINATED | -1 | X'FFFFFFFF' |
| MQVL_EMPTY_STRING | 0 | X'00000000' |

MQVU_* (变量用户标识)

| 表 408: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQVU_FIXED_USER | 1 | X'00000001' |
| MQVU_ANY_USER | 2 | X'00000002' |

MQWDR_* (集群工作负载出口目标记录结构)

| 表 409: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQWDR_STRUC_ID | "WDR↵" |
| MQWDR_STRUC_ID_ARRAY | 'W','D','R','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 410: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (等待时间间隔)

| 表 411: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQWI_UNLIMITED | -1 | X'FFFFFFFF' |

MQWIH_* (工作负载信息头结构和标志)

工作负载信息头结构

| 表 412: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQWIH_STRUC_ID | "WIH↵" |
| MQWIH_STRUC_ID_ARRAY | 'W','I','H','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 413: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQWIH_VERSION_1 | 1 | X'00000001' |
| MQWIH_CURRENT_VERSION | 1 | X'00000001' |
| MQWIH_LENGTH_1 | 120 | X'00000078' |
| MQWIH_CURRENT_LENGTH | 120 | X'00000078' |

工作负载信息头标志

| 表 414: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQWIH_NONE | 0 | X'00000000' |

MQWQR_* (集群工作负载出口队列记录结构)

| 表 415: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQWQR_STRUC_ID | "WQR↵" |
| MQWQR_STRUC_ID_ARRAY | 'W','Q','R','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 416: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (通配符模式)

| 表 417: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQWS_DEFAULT | 0 | X'00000000' |
| MQWS_CHAR | 1 | X'00000001' |
| MQWS_TOPIC | 2 | X'00000002' |

MQWXP_* (集群工作负载出口参数结构)

MQWXP_* (集群工作负载出口参数结构)

| 表 418: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQWXP_STRUC_ID | "WXP↵" |
| MQWXP_STRUC_ID_ARRAY | 'W','X','P','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 419: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQWXP_VERSION_1 | 1 | X'00000001' |

| 表 419: 常量值 (继续) | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (集群工作负载标志)

| 表 420: 常量值 | | |
|--------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQWXP_PUT_BY_CLUSTER_CHL | 2 | X'00000002' |

相关参考

MQWXP 中的字段-集群工作负载出口参数结构

MQXACT_* (API 调用者类型)

| 表 421: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXACT_EXTERNAL | 1 | X'00000001' |
| MQXACT_INTERNAL | 2 | X'00000002' |

MQXC_* (出口命令)

| 表 422: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (出口响应)

| 表 423: 常量值 | | |
|--------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

| 表 423: 常量值 (继续) | | |
|---------------------|-----|--------------|
| : | 小数值 | 十六进制值 |
| MQXCC_SUPPRESS_EXIT | -5 | X'FFFFFFFFB' |
| MQXCC_CLOSE_CHANNEL | -6 | X'FFFFFFFA' |
| MQXCC_REQUEST_ACK | -7 | X'FFFFFFF9' |
| MQXCC_FAILED | -8 | X'FFFFFFF8' |

MQXDR_* (出口响应)

| 表 424: 常量值 | | |
|-------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXDR_OK | 0 | X'00000000' |
| MQXDR_CONVERSION_FAILED | 1 | X'00000001' |

MQXE_* (环境)

| 表 425: 常量值 | | |
|---------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (注册入口点选项结构和出口选项)

"注册入口点选项" 结构

| 表 426: 常量结构 | |
|-----------------------|--------------------|
| : | 结构 |
| MQXEPO_STRUC_ID | "XEPO" |
| MQXEPO_STRUC_ID_ARRAY | 'X', 'E', 'P', 'O' |

注: 符号 – 表示单个空白字符。

| 表 427: 常量值 | | |
|------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXEPO_VERSION_1 | 1 | X'00000001' |
| MQXEPO_CURRENT_VERSION | 1 | X'00000001' |

退出选项

| 表 428: 常量值 | | |
|-------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXEPO_NONE | 0 | X'00000000' |

MQXF_* (API 函数标识)

| 表 429: 常量值 | | |
|-----------------------|------|-------------|
| : | 小数制值 | 十六进制值 |
| 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' |

MQXP_* (API 交叉出口参数结构)

| 表 430: 常量结构 | |
|---------------------|--------------------|
| : | 结构 |
| MQXP_STRUC_ID | "XP↵" |
| MQXP_STRUC_ID_ARRAY | 'X', 'P', '↵', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 431: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXP_VERSION_1 | 1 | X'00000001' |

MQXPDA_* (问题确定区域)

| 表 432: 常量名称和值 | |
|-------------------|--------------------------|
| : | 值 |
| MQXPDA_NONE | X'00...00' (48 个空值) |
| MQXPDA_NONE_ARRAY | '\0', '\0', ... (48 个空值) |

MQXPT_* (传输类型)

| 表 433: 常量值 | | |
|---------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (传输队列头结构)

| 表 434: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQXQH_STRUC_ID | "XQH↵" |
| MQXQH_STRUC_ID_ARRAY | 'X', 'Q', 'H', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 435: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXQH_VERSION_1 | 1 | X'00000001' |
| MQXQH_CURRENT_VERSION | 1 | X'00000001' |

MQXR_* (退出原因)

| 表 436: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_PARS | 29 | X'0000001D' |

MQXR2_* (出口响应 2)

| 表 437: 常量值 | | |
|----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXR2_PUT_WITH_DEF_ACTION | 0 | X'00000000' |
| MQXR2_PUT_WITH_DEF_USERID | 1 | X'00000001' |
| MQXR2_PUT_WITH_MSG_USERID | 2 | X'00000002' |
| MQXR2_USE_AGENT_BUFFER | 0 | X'00000000' |
| MQXR2_USE_EXIT_BUFFER | 4 | X'00000004' |
| MQXR2_DEFAULT_CONTINUATION | 0 | X'00000000' |
| MQXR2_CONTINUE_CHAIN | 8 | X'00000008' |
| MQXR2_SUPPRESS_CHAIN | 16 | X'00000010' |
| MQXR2_STATIC_CACHE | 0 | X'00000000' |
| MQXR2_DYNAMIC_CACHE | 32 | X'00000020' |

MQXT_* (出口标识)

| 表 438: 常量值 | | |
|-----------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (出口用户区域值)

| 表 439: 常量名称和值 | |
|------------------|--------------------------|
| : | 值 |
| MQXUA_NONE | X'00...00' (16 个空值) |
| MQXUA_NONE_ARRAY | '\0', '\0', ... (16 个空值) |

MQXWD_* (出口等待描述符结构)

| 表 440: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQXWD_STRUC_ID | "XWD↵" |
| MQXWD_STRUC_ID_ARRAY | 'X', 'W', 'D', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 441: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQXWD_VERSION_1 | 1 | X'00000001' |

MQZAC_* (应用程序上下文结构)

| 表 442: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQZAC_STRUC_ID | "ZAC↵" |
| MQZAC_STRUC_ID_ARRAY | 'Z', 'A', 'C', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 443: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZAC_VERSION_1 | 1 | X'00000001' |
| MQZAC_CURRENT_VERSION | 1 | X'00000001' |

MQZAD_* (权限数据结构)

| 表 444: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQZAD_STRUC_ID | "ZAD↵" |
| MQZAD_STRUC_ID_ARRAY | 'Z','A','D','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 445: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZAD_VERSION_1 | 1 | X'00000001' |
| MQZAD_VERSION_2 | 2 | X'00000002' |
| MQZAD_CURRENT_VERSION | 2 | X'00000002' |

MQZAET_* (可安装服务实体类型)

| 表 446: 常量值 | | |
|------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZAET_NONE | 0 | X'00000000' |
| MQZAET_PRINCIPAL | 1 | X'00000001' |
| MQZAET_GROUP | 2 | X'00000002' |
| MQZAET_UNKNOWN | 3 | X'00000003' |

MQZAO_* (可安装服务授权)

| 表 447: 常量值 | | |
|--------------------------------|--------|-------------|
| : | 小数值 | 十六进制值 |
| MQZAO_CONNECT | 1 | X'00000001' |
| MQZAO_BROWSE | 2 | X'00000002' |
| MQZAO_INPUT | 4 | X'00000004' |
| MQZAO_OUTPUT | 8 | X'00000008' |
| MQZAO_INQUIRE | 16 | X'00000010' |
| MQZAO_SET | 32 | X'00000020' |
| MQZAO_PASS_IDENTITY_CONTEXT | 64 | X'00000040' |
| MQZAO_PASS_ALL_CONTEXT | 128 | X'00000080' |
| MQZAO_SET_IDENTITY_CONTEXT | 256 | X'00000100' |
| MQZAO_SET_ALL_CONTEXT | 512 | X'00000200' |
| MQZAO_ALTERNATE_USER_AUTHORITY | 1024 | X'00000400' |
| MQZAO_PUBLISH | 2048 | X'00000800' |
| MQZAO_SUBSCRIBE | 4096 | X'00001000' |
| MQZAO_RESUME | 8192 | X'00002000' |
| MQZAO_ALL_MQI | 16383 | X'00003FFF' |
| MQZAO_CREATE | 65536 | X'00010000' |
| MQZAO_DELETE | 131072 | X'00020000' |
| MQZAO_DISPLAY | 262144 | X'00040000' |

| 表 447: 常量值 (继续) | | |
|------------------------|----------|-------------|
| : | 小数值 | 十六进制值 |
| MQZAO_CHANGE | 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_* (可安装服务接口版本)

| 表 448: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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_* (认证类型)

| 表 449: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZAT_INITIAL_CONTEXT | 0 | X'00000000' |
| MQZAT_CHANGE_CONTEXT | 1 | X'00000001' |

MQZCI_* (可安装服务延续指示符)

| 表 450: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZCI_DEFAULT | 0 | X'00000000' |
| MQZCI_CONTINUE | 0 | X'00000000' |
| MQZCI_STOP | 1 | X'00000001' |

MQZED_* (实体数据结构)

| 表 451: 常量结构 | |
|----------------------|--------------------|
| : | 结构 |
| MQZED_STRUC_ID | "ZED↵" |
| MQZED_STRUC_ID_ARRAY | 'Z', 'E', 'D', '↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 452: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZED_VERSION_1 | 1 | X'00000001' |
| MQZED_VERSION_2 | 2 | X'00000002' |
| MQZED_CURRENT_VERSION | 2 | X'00000002' |

MQZFP_* (可用参数结构)

| 表 453: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQZFP_STRUC_ID | "ZFP↵" |
| MQZFP_STRUC_ID_ARRAY | 'Z','F','P','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 454: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZFP_VERSION_1 | 1 | X'00000001' |
| MQZFP_CURRENT_VERSION | 1 | X'00000001' |

MQZIC_* (身份上下文结构)

| 表 455: 常量结构 | |
|----------------------|-----------------|
| : | 结构 |
| MQZIC_STRUC_ID | "ZIC↵" |
| MQZIC_STRUC_ID_ARRAY | 'Z','I','C','↵' |

注: 符号 ↵ 表示单个空白字符。

| 表 456: 常量值 | | |
|-----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZIC_VERSION_1 | 1 | X'00000001' |
| MQZIC_CURRENT_VERSION | 1 | X'00000001' |

MQZID_* (服务的功能标识)

所有服务的公共功能标识

| 表 457: 常量值 | | |
|------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZID_INIT | 0 | X'00000000' |
| MQZID_TERM | 1 | X'00000001' |

"权限" 服务的功能标识

| 表 458: 常量值 | | |
|----------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZID_INIT_AUTHORITY | 0 | X'00000000' |

| 表 458: 常量值 (继续) | | |
|--------------------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

名称服务的函数标识

| 表 459: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| 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' |

用户标识服务的函数标识

| 表 460: 常量值 | | |
|-------------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZID_INIT_USERID | 0 | X'00000000' |
| MQZID_TERM_USERID | 1 | X'00000001' |
| MQZID_FIND_USERID | 2 | X'00000002' |

MQZIO_* (可安装服务初始化选项)

| 表 461: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZIO_PRIMARY | 0 | X'00000000' |
| MQZIO_SECONDARY | 1 | X'00000001' |

MQZNS_* (名称服务接口版本)

| 表 462: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZNS_VERSION_1 | 1 | X'00000001' |

MQZSE_* (可安装服务启动-枚举指示符)

| 表 463: 常量值 | | |
|----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZSE_START | 1 | X'00000001' |
| MQZSE_CONTINUE | 0 | X'00000000' |

MQZSL_* (可安装服务选择器指示符)

| 表 464: 常量值 | | |
|--------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZSL_NOT_返回 | 0 | X'00000000' |
| 已返回 MQZSL_退回 | 1 | X'00000001' |

MQZTO_* (可安装服务终止选项)

| 表 465: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZTO_PRIMARY | 0 | X'00000000' |
| MQZTO_SECONDARY | 1 | X'00000001' |

MQZUS_* (用户标识服务接口版本)

| 表 466: 常量值 | | |
|-----------------|-----|-------------|
| : | 小数值 | 十六进制值 |
| MQZUS_VERSION_1 | 1 | X'00000001' |

MQI 中使用的数据类型

有关可在 MQI 中使用的数据类型的信息。每种数据类型的相关语言的描述，字段和语言声明。

介绍 MQI 中使用的数据类型

本部分介绍了 MQI 中使用的数据类型，并为您提供了一些有关在受支持的编程语言中使用这些数据类型的指导。

基本数据类型

本部分包含有关 MQI (或出口函数) 中使用的数据类型的信息。下面详细描述了这些内容，然后在以下主题中显示了如何在受支持的编程语言中声明基本数据类型的示例。

MQI (或出口函数) 中使用的数据类型为:

- 基本数据类型，或
- 基本数据类型 (数组或结构) 的聚集

MQI (或出口函数) 中使用以下基本数据类型:

| 基本数据类型名称 | 数据类型 | 描述 |
|----------|-----------------|---|
| MQBOOL | 布尔 | MQBOOL 数据类型表示布尔值。值 0 表示 false。任何其他值都表示 true。MQBOOL 必须与 MQLONG 数据类型一致。 |
| MQBYTE | Byte | <p>MQBYTE 数据类型表示单个字节的数据。没有特殊解释放在字节上; 它被视为一串位, 而不被视为二进制数字或字符。不需要特殊对齐。</p> <p>在使用不同字符集或编码的队列管理器之间发送 MQBYTE 数据时, 不会以任何方式转换 MQBYTE 数据。MQMD 结构中的 <i>MsgId</i> 和 <i>CorrelId</i> 字段如下所示。</p> <p>MQBYTE 的数组有时用于表示队列管理器未知的主存储器区域。例如, 该区域可能包含应用程序消息数据或结构。此区域的边界对齐必须与其中包含的数据的性质兼容。</p> <p>在 C 编程语言中, 任何数据类型都可以用于显示为 MQBYTE 数组的函数参数。这是因为此类参数总是由地址传递, 而在 C 中, 函数参数被声明为指针到空。</p> |
| MQBYTEn | <i>n</i> 字节的字符串 | <p>每种 MQBYTEn 数据类型都表示一个 <i>n</i> 字节的字符串, 其中 <i>n</i> 可以采用以下任何值: 8, 16, 24, 32, 40 或 128。每个字节都由 MQBYTE 数据类型描述。不需要特殊对齐。</p> <p>如果字节字符串中的数据比定义的字符串长度短, 那么必须使用空值填充数据以填充字符串。</p> <p>当队列管理器将字节字符串返回到应用程序 (例如, 在 MQGET 调用上) 时, 队列管理器会将空值填充到定义的字符串长度。</p> <p>指定的常量可用于定义字节字符串字段的长度。这些内容在 第 49 页的『常量』 中列出</p> |

| 基本数据类型名称 | 数据类型 | 描述 |
|-----------|-----------|--|
| MQCHAR | 字符 | <p>MQCHAR 数据类型表示单字节字符，或者表示双字节或多字节字符的一个字节。不需要特殊对齐。</p> <p>在使用不同字符集或编码的队列管理器之间发送 MQCHAR 数据时，MQCHAR 数据通常需要转换才能正确解释数据。队列管理器会自动对 MQMD 结构中的 MQCHAR 数据执行此操作。应用程序消息数据中 MQCHAR 数据的转换由 MQGET 调用上指定的 MQGMO_CONVERT 选项控制；请参阅第 324 页的『MQGMO-Get-消息选项』中此选项的描述以获取更多详细信息。</p> |
| MQCHARn | n 个字符的字符串 | <p>每种 MQCHARn 数据类型都表示一个由 n 个字符组成的字符串，其中 n 可以采用以下任何值：4，8，12，20，28，32，48，64，128 或 256。每个字符都由 MQCHAR 数据类型描述。不需要特殊对齐。</p> <p>如果字符串中的数据短于定义的字符串长度，那么必须用空格填充数据以填充字符串。在某些情况下，可以使用空字符来过早结束字符串，而不是用空格填充；空字符及其后面的字符被视为空格，直至字符串的定义长度。可以使用空值的位置在调用和数据类型描述中标识。</p> <p>当队列管理器将字符串返回到应用程序（例如，在 MQGET 调用上）时，队列管理器始终以空白填充字符串的定义长度；队列管理器不会使用空字符来定界字符串。</p> <p>指定的常量可用于定义字符串字段的长度，并在第 49 页的『常量』中列出。</p> |
| MQFLOAT32 | 32 位浮点数 | <p>MQFLOAT32 数据类型是使用标准 IEEE 浮点格式表示的 32 位浮点数。MQFLOAT32 必须在 4 字节边界上对齐。</p> <p>在 z/OS 上的 C 中使用 MQFLOAT32 需要使用 FLOAT (IEEE) 编译器标志。</p> <p>在 COBOL 中使用 MQFLOAT32 仅限于支持 IEEE 格式的浮点数的编译器。这可能需要使用 FLOAT (NATIVE) 编译器标志。</p> |

| 基本数据类型名称 | 数据类型 | 描述 |
|-----------|---------|---|
| MQFLOAT64 | 64 位浮点数 | <p>MQFLOAT64 数据类型是使用标准 IEEE 浮点格式表示的 64 位浮点数。MQFLOAT64 必须在 8 字节边界上对齐。</p> <p>在 z/OS 上的 C 中使用 MQFLOAT64 需要使用 FLOAT (IEEE) 编译器标志。</p> <p>在 COBOL 中使用 MQFLOAT64 仅限于支持 IEEE 格式的浮点数的编译器。这可能需要使用 FLOAT (NATIVE) 编译器标志。</p> |
| MQHCONFIG | 配置句柄 | <p>MQHCONFIG 数据类型表示配置句柄，即正在为特定可安装服务配置的组件。配置句柄必须在其自然边界上对齐。</p> <p>应用程序不得依赖于此句柄中存储的数据的格式。如果有效，那么其值将在进一步的 MQI 调用中可用，但不打算具有除此用途以外的任何含义。</p> |
| MQHCONN | 连接句柄 | <p>MQHCONN 数据类型表示连接句柄，即与特定队列管理器的连接。连接句柄必须在 4 字节边界上对齐。</p> <p>应用程序不得依赖于此句柄中存储的数据的格式。如果有效，那么其值将在进一步的 MQI 调用中可用，但不打算具有除此用途以外的任何含义。</p> |
| MQHMSG | 消息句柄 | <p>MQHMSG 数据类型表示允许访问消息的消息句柄。消息句柄必须在 8 字节边界上对齐。</p> <p>应用程序不得依赖于此句柄中存储的数据的格式。如果有效，那么其值将在进一步的 MQI 调用中可用，但不打算具有除此用途以外的任何含义。</p> |
| MQHOBJ | 对象句柄 | <p>MQHOBJ 数据类型表示授予对象访问权的对象句柄。对象句柄必须在 4 字节边界上对齐。</p> <p>应用程序不得依赖于此句柄中存储的数据的格式。如果有效，那么其值将在进一步的 MQI 调用中可用，但不打算具有除此用途以外的任何含义。</p> |

| 基本数据类型名称 | 数据类型 | 描述 |
|----------|-----------|---|
| MQINT8 | 8 位带符号整数 | MQINT8 数据类型是 8 位带符号整数，可采用范围在 -128 到 +127 之间的任何值，除非上下文另有限制。 |
| MQINT16 | 16 位带符号整数 | MQINT16 数据类型是一个 16 位带符号整数，可以采用 -32 768 到 +32 767 范围内的任何值，除非上下文另有限制。MQINT16 必须在 2 字节边界上对齐。 |
| MQINT32 | 32 位带符号整数 | MQINT32 数据类型是 32 位带符号二进制整数，可采用范围在 -2 147 483 648 到 + 2 147 483 647 之间的任何值，除非上下文另有限制。 请参阅 MQLONG 的定义。 |
| MQINT64 | 64 位带符号整数 | MQINT64 数据类型是 64 位带符号整数，可以采用范围在 -9 223 372 036 854 775 808 到 + 9 223 372 036 854 775 807 之间的任何值，除非上下文另有限制。 对于 COBOL，有效范围限制为 -999 999 999 999 999 999 到 +999 999 999 999 999 999。64 位整数必须在 8 字节边界上对齐。 |
| MQLONG | 32 位带符号整数 | MQLONG 数据类型是 32 位有符号的二进制整数，可采用范围在 -2 147 483 648 到 + 2 147 483 647 之间的任何值，除非上下文另有限制。 对于 COBOL，有效范围限制为 -999 999 999 到 +999 999 999。MQLONG 必须在 4 字节边界上对齐。 |
| MQPID | 进程标识 | WebSphere MQ 进程标识。 这是 MQ 跟踪和 FFST™ 转储中使用的相同标识，但可能与操作系统进程标识不同。 |
| MQPTR | 指针 | MQPTR 数据类型是任何类型的数据的地址。指针必须在其自然边界上对齐；这是 IBM i 上的 16 字节边界，在其他平台上为 8 字节边界。 某些编程语言支持有类型的指针；MQI 也在一些情况下使用这些指针（例如，在 C 编程语言中使用 PMQCHAR 和 PMQLONG）。 |

| 基本数据类型名称 | 数据类型 | 描述 |
|----------|-----------|--|
| MQTID | 线程标识 | WebSphere MQ 线程标识。 这是 MQ 跟踪和 FFST™ 转储中使用的相同标识，但可能与操作系统线程标识不同。 |
| MQUINT8 | 8 位无符号整数 | MQUINT8 数据类型是一个 8 位无符号整数，可采用范围在 0 到 +255 之间的任何值，除非上下文另有限制。 |
| MQUINT16 | 16 位无符号整数 | MQUINT16 数据类型是一个 16 位无符号整数，可以采用 0 到 +65 535 范围内的任何值，除非上下文另有限制。MQUINT16 必须在 2 字节边界上对齐。 |
| MQUINT32 | 32 位无符号整数 | MQUINT32 数据类型是 32 位无符号二进制整数。 请参阅 MQULONG 的定义。 |
| MQUINT64 | 64 位无符号整数 | MQUINT64 数据类型是 64 位无符号整数，可以采用 0 到 +18 446 744 073 709 551 615 范围内的任何值，除非上下文另有限制。 对于 COBOL，有效范围限制为 0 到 +999 999 999 999 999 999。64 位整数必须在 8 字节边界上对齐。 |
| MQULONG | 32 位无符号整数 | MQULONG 数据类型是 32 位无符号二进制整数，可以采用 0 到 +4 294 967 294 范围内的任何值，除非上下文另有限制。 对于 COBOL，有效范围限制为 0 到 +999 999 999。MQULONG 必须在 4 字节边界上对齐。 |
| PMQACH | 指针 | 指向 MQACH 类型的数据结构的指针 |
| PMQAIR | 指针 | 指向 MQAIR 类型的数据结构的指针 |
| PMQAXC | 指针 | 指向 MQAXC 类型的数据结构的指针 |
| PMQAXP | 指针 | 指向 MQAXP 类型的数据结构的指针 |
| PMQBMHO | 指针 | 指向 MQBMHO 类型的数据结构的指针 |
| PMQBO | 指针 | 指向 MQBO 类型的数据结构的指针 |
| PMQBOOL | 指针 | 指向 MQBOOL 类型的数据的指针 |

| 基本数据类型名称 | 数据类型 | 描述 |
|------------|------|---|
| PMQBYTE | 指针 | 指向 MQBYTE 类型数据的指针 |
| PMQBYTEn | 指针 | 指向类型为 MQBYTEn 的数据的指针，其中 n 可以是 8, 16,24,32,40,128 |
| PMQCBC | 指针 | 指向 MQCBC 类型的数据结构的指针 |
| PMQCBD | 指针 | 指向 MQCBD 类型的数据结构的指针 |
| PMQCHAR | 指针 | 指向 MQCHAR 类型的数据的指针 |
| PMQCHARN | 指针 | 指向数据类型 MQCHARN 的指针，其中 n 可以是 4, 8, 12, 20, 28, 32, 48, 64, 128, 256, 264 |
| PMQCHARV | 指针 | 指向 MQCHARV 类型的数据结构的指针 |
| PMQCIH | 指针 | 指向 MQCIH 类型的数据结构的指针 |
| PMQCMHO | 指针 | 指向 MQCMHO 类型的数据结构的指针 |
| PMQCNO | 指针 | 指向 MQCNO 类型的数据结构的指针 |
| PMQCSP | 指针 | 指向 MQCSP 类型的数据结构的指针 |
| PMQCTLO | 指针 | 指向 MQCTLO 类型的数据结构的指针 |
| PMQDHO | 指针 | 指向 MQDHO 类型的数据结构的指针 |
| PMQDLH | 指针 | 指向 MQDLH 类型的数据结构的指针 |
| PMQDMHO | 指针 | 指向 MQDMHO 类型的数据结构的指针 |
| PMQDMPO | 指针 | 指向 MQDMPO 类型的数据结构的指针 |
| PMQEPH | 指针 | 指向 MQEPH 类型的数据结构的指针 |
| PMQFLOAT32 | 指针 | 指向类型为 MQFLOAT32 的数据结构的指针 |
| PMQFLOAT64 | 指针 | 指向类型为 MQFLOAT64 的数据结构的指针 |
| PMQFUNC | 指针 | 指向函数的指针 |

| 基本数据类型名称 | 数据类型 | 描述 |
|------------|------|-----------------------|
| PMQGM0 | 指针 | 指向 MQGM0 类型的数据结构的指针 |
| PMQHCONFIG | 指针 | 指向 MQHCONFIG 类型的数据的指针 |
| PMQHCONN | 指针 | 指向 MQHCONN 类型的数据的指针 |
| PMQHMSG | 指针 | 指向 MQHMSG 类型数据的指针 |
| PMQHOBJ | 指针 | 指向 MQHOBJ 类型的数据的指针 |
| PMQIIH | 指针 | 指向 MQIIH 类型的数据结构的指针 |
| PMQIMPO | 指针 | 指向 MQIMPO 类型的数据结构的指针 |
| PMQINT8 | 指针 | 指向类型为 MQINT8 的数据的指针 |
| PMQINT16 | 指针 | 指向类型为 MQINT16 的数据的指针 |
| PMQINT32 | 指针 | 指向类型为 MQINT32 的数据的指针 |
| PMQINT64 | 指针 | 指向类型为 MQINT64 的数据的指针 |
| PMQLONG | 指针 | 指向 MQLONG 类型的数据的指针 |
| PMQMD | 指针 | 指向 MQMD 类型的结构的指针 |
| PMQMDE | 指针 | 指向 MQMDE 类型的数据结构的指针 |
| PMQMD1 | 指针 | 指向类型为 MQMD1 的数据结构的指针 |
| PMQMD2 | 指针 | 指向类型为 MQMD2 的数据结构的指针 |
| PMQMHBO | 指针 | 指向 MQMHBO 类型的数据结构的指针 |
| PMQOD | 指针 | 指向 MQOD 类型的数据结构的指针 |
| PMQOR | 指针 | 指向 MQOR 类型的数据结构的指针 |
| PMQPD | 指针 | 指向 MQPD 类型的数据结构的指针 |
| PMQPID | 指针 | 指向进程标识的指针 |
| PMQMD | 指针 | 指向 MQMD 类型的数据结构的指针 |
| PMQPMO | 指针 | 指向 MQPMO 类型的数据结构的指针 |
| PMQPTR | 指针 | 指向 MQPTR 类型的数据的指针 |

| 基本数据类型名称 | 数据类型 | 描述 |
|-----------|------|-----------------------|
| PMQRFH | 指针 | 指向 MQRFH 类型的数据结构的指针 |
| PMQRFH2 | 指针 | 指向类型为 MQRFH2 的数据结构的指针 |
| PMQRMH | 指针 | 指向 MQRMH 类型的数据结构的指针 |
| PMQRR | 指针 | 指向 MQRR 类型的数据结构的指针 |
| PMQSCO | 指针 | 指向 MQSCO 类型的数据结构的指针 |
| PMQSD | 指针 | 指向 MQSD 类型的数据结构的指针 |
| PMQSMPO | 指针 | 指向 MQSMPO 类型的数据结构的指针 |
| PMQSRO | 指针 | 指向 MQSRO 类型的数据结构的指针 |
| PMSSTS | 指针 | 指向 MQSTS 类型的数据结构的指针 |
| PMQTID | 指针 | 指向线程标识的指针 |
| PMQTM | 指针 | 指向 MQTM 类型的数据结构的指针 |
| PMQTC2 | 指针 | 指向类型为 MQTC2 的数据结构的指针 |
| PMQUINT8 | 指针 | 指向数据类型 MQUINT8 的指针 |
| PMQUINT16 | 指针 | 指向数据类型 MQUINT16 的指针 |
| PMQUINT32 | 指针 | 指向数据类型 MQUINT32 的指针 |
| PMQUINT64 | 指针 | 指向数据类型 MQUINT64 的指针 |
| PMQULONG | 指针 | 指向数据类型 MQULONG 的指针 |
| PMQVOID | 指针 | |
| PMQWIH | 指针 | 指向 MQWIH 类型的数据结构的指针 |
| PMQXQH | 指针 | 指向 MQXQH 类型的数据结构的指针 |

C 声明

| 数据类型 | 表示 |
|---------|--|
| MQBOOL | <pre>typedef MQLONG MQBOOL;</pre> |
| MQBYTE | <pre>typedef unsigned char MQBYTE;</pre> |
| MQBYTE8 | <pre>typedef MQBYTE MQBYTE8[8];</pre> |

| 数据类型 | 表示 |
|-----------|--|
| MQBYTE16 | <code>typedef MQBYTE MQBYTE16[16];</code> |
| MQBYTE24 | <code>typedef MQBYTE MQBYTE24[24];</code> |
| MQBYTE32 | <code>typedef MQBYTE MQBYTE32[32];</code> |
| MQBYTE40 | <code>typedef MQBYTE MQBYTE40[40];</code> |
| MQCHAR | <code>typedef char MQCHAR;</code> |
| MQCHAR4 | <code>typedef MQCHAR MQCHAR4[4];</code> |
| MQCHAR8 | <code>typedef MQCHAR MQCHAR8[8];</code> |
| MQCHAR12 | <code>typedef MQCHAR MQCHAR12[12];</code> |
| MQCHAR20 | <code>typedef MQCHAR MQCHAR20[20];</code> |
| MQCHAR28 | <code>typedef MQCHAR MQCHAR28[28];</code> |
| MQCHAR32 | <code>typedef MQCHAR MQCHAR32[32];</code> |
| MQCHAR48 | <code>typedef MQCHAR MQCHAR48[48];</code> |
| MQCHAR64 | <code>typedef MQCHAR MQCHAR64[64];</code> |
| MQCHAR128 | <code>typedef MQCHAR MQCHAR128[128];</code> |
| MQCHAR256 | <code>typedef MQCHAR MQCHAR256[256];</code> |
| MQFLOAT32 | <code>typedef float MQFLOAT32;</code> |
| MQFLOAT64 | <code>typedef double MQFLOAT64;</code> |
| MQHCONFIG | <code>typedef void MQPOINTER MQHCONFIG;</code> |
| MQHCONN | <code>typedef MQLONG MQHCONN;</code> |
| MQHOBJ | <code>typedef MQLONG MQHOBJ;</code> |
| MQINT8 | <code>typedef signed char MQINT8;</code> |

| 数据类型 | 表示 |
|----------|---|
| MQINT16 | <pre>typedef short MQINT16;</pre> |
| MQINT64 | <p>在 64 位 UNIX 系统上:</p> <pre>typedef long;</pre> <p>在 32 位 AIX, Solaris 和 HP-UX 上:</p> <pre>typedef int64_t;</pre> <p>在 IBM i, Linux 和 z/OS 上:</p> <pre>typedef long long;</pre> <p>在 Windows 上:</p> <pre>typedef _int64;</pre> |
| MQLONG | <p>在 IBM i 上:</p> <pre>typedef long MQLONG;</pre> <p>其他平台:</p> <pre>if defined(MQ_64_BIT) typedef int MQLONG; else typedef long MQLONG;</pre> |
| MQPID | <pre>typedef MQLONG MQPID;</pre> |
| MQPTR | <pre>typedef void MQPOINTER MQPTR;</pre> |
| MQTID | <pre>typedef MQLONG MQTID;</pre> |
| MQUINT8 | <pre>typedef unsigned char MQUINT8;</pre> |
| MQUINT16 | <pre>typedef unsigned short MQUINT16;</pre> |
| MQUINT64 | <p>在 64 位 UNIX 系统上:</p> <pre>typedef unsigned long;</pre> <p>在 32 位 AIX, Solaris 和 HP-UX 上:</p> <pre>typedef uint64_t;</pre> <p>在 IBM i, Linux 和 z/OS 上:</p> <pre>typedef unsigned long long;</pre> <p>在 Windows 上:</p> <pre>typedef unsigned _int64;</pre> |

| 数据类型 | 表示 |
|------------|---|
| MQULONG | 在 IBM i 上: <pre>typedef unsigned long MQULONG;</pre> 其他平台: <pre>if defined(MQ_64_BIT) typedef unsigned int MQULONG; else typedef unsigned long MQULONG;</pre> |
| PMQBO | <pre>typedef MQBO MQPOINTER PMQBO;</pre> |
| PMQBOOL | <pre>typedef MQBOOL MQPOINTER PMQBOOL;</pre> |
| PMQBYTE | <pre>typedef MQBYTE MQPOINTER PMQBYTE;</pre> |
| PMQBYTE8 | <pre>typedef MQBYTE8[8] MQPOINTER PMQBYTE8[8];</pre> |
| PMQBYTE16 | <pre>typedef MQBYTE16[16] MQPOINTER PMQBYTE16[16];</pre> |
| PMQBYTE24 | <pre>typedef MQBYTE24[24] MQPOINTER PMQBYTE24[24];</pre> |
| PMQBYTE32 | <pre>typedef MQBYTE32[32] MQPOINTER PMQBYTE32[32];</pre> |
| PMQBYTE40 | <pre>typedef MQBYTE40[40] MQPOINTER PMQBYTE40[40];</pre> |
| PMQBYTE128 | <pre>typedef MQBYTE128[128] MQPOINTER PMQBYTE128[128];</pre> |
| PMQCHAR | <pre>typedef MQCHAR MQPOINTER PMQCHAR;</pre> |
| PMQCHAR4 | <pre>typedef MQCHAR4[4] MQPOINTER PMQCHAR4[4];</pre> |
| PMQCHAR8 | <pre>typedef MQCHAR8[8] MQPOINTER PMQCHAR8[8];</pre> |
| PMQCHAR12 | <pre>typedef MQCHAR12[12] MQPOINTER PMQCHAR12[12];</pre> |
| PMQCHAR20 | <pre>typedef MQCHAR20[20] MQPOINTER PMQCHAR20[20];</pre> |
| PMQCHAR28 | <pre>typedef MQCHAR28[28] MQPOINTER PMQCHAR28[28];</pre> |
| PMQCHAR32 | <pre>typedef MQCHAR32[32] MQPOINTER PMQCHAR32[32];</pre> |
| PMQCHAR48 | <pre>typedef MQCHAR48[48] MQPOINTER PMQCHAR48[48];</pre> |
| PMQCHAR64 | <pre>typedef MQCHAR64[64] MQPOINTER PMQCHAR64[64];</pre> |

| 数据类型 | 表示 |
|------------|--|
| PMQCHAR128 | <code>typedef MQCHAR128[128] MQPOINTER PMQCHAR128[128];</code> |
| PMQCHAR256 | <code>typedef MQCHAR256[256] MQPOINTER PMQCHAR256[256];</code> |
| PMQCHAR264 | <code>typedef MQCHAR264[264] MQPOINTER PMQCHAR264[264];</code> |
| PMQCIH | <code>typedef MQCIH MQPOINTER PMQCIH;</code> |
| PMQCNO | <code>typedef MQCNO MQPOINTER PMQCNO;</code> |
| PMQDLH | <code>typedef MQDLH MQPOINTER PMQDLH;</code> |
| PMQFUNC | <code>typedef void MQPOINTER PMQFUNC;</code> |
| PMQFLOAT32 | <code>typedef MQFLOAT32 MQPOINTER PMQFLOAT32;</code> |
| PMQFLOAT64 | <code>typedef MQFLOAT64 MQPOINTER PMQFLOAT64;</code> |
| PMQGMO | <code>typedef MQGMO MQPOINTER PMQGMO;</code> |
| PMQHCONFIG | <code>typedef MQHCONFIG MQPOINTER PMQHCONFIG;</code> |
| PMQHCONN | <code>typedef MQHCONN MQPOINTER PMQHCONN;</code> |
| PMQHOBJ | <code>typedef MQHOBJ MQPOINTER PMQHOBJ;</code> |
| PMQIIH | <code>typedef MQIIH MQPOINTER PMQIIH;</code> |
| PMQINT8 | <code>typedef MQINT8 MQPOINTER PMQINT8;</code> |
| PMQINT16 | <code>typedef MQINT16 MQPOINTER PMQINT16;</code> |
| PMQLONG | <code>typedef MQLONG MQPOINTER PMQLONG;</code> |
| PMQMD | <code>typedef MQMD MQPOINTER PMQMD;</code> |
| PMQMD1 | <code>typedef MQMD1[1] MQPOINTER PMQMD1[1];</code> |
| PMQMDE | <code>typedef MQMDE MQPOINTER PMQMDE;</code> |
| PMQOD | <code>typedef MQOD MQPOINTER PMQOD;</code> |

| 数据类型 | 表示 |
|-----------|--|
| PMQPMO | <code>typedef MQPMO MQPOINTER PMQPMO;</code> |
| PMQPTR | <code>typedef MQPTR MQPOINTER PMQPTR;</code> |
| PMQRFH | <code>typedef MQRFH MQPOINTER PMQRFH;</code> |
| PMQRFH2 | <code>typedef MQRFH2[2] MQPOINTER PMQRFH2[2];</code> |
| PMQRMH | <code>typedef MQRMH MQPOINTER PMQRMH;</code> |
| PMQTM | <code>typedef MQTM MQPOINTER PMQTM;</code> |
| PMQTM2 | <code>typedef MQTM2[2] MQPOINTER PMQTM2[2];</code> |
| PMQUINT8 | <code>typedef MQUINT8 MQPOINTER PMQUINT8;</code> |
| PMQUINT16 | <code>typedef MQUINT16 MQPOINTER PMQUINT16;</code> |
| PMQULONG | <code>typedef MQULONG MQPOINTER PMQULONG;</code> |
| PMQVOID | <code>typedef void MQPOINTER PMQVOID;</code> |
| PMQWIH | <code>typedef MQWIH MQPOINTER PMQWIH;</code> |
| PMQXQH | <code>typedef MQXQH MQPOINTER PMQXQH;</code> |
| Ppmqbo | <code>typedef PMQBO MQPOINTER PPMQBO;</code> |
| PPMQBYTE | <code>typedef PMQBYTE MQPOINTER PPMQBYTE;</code> |
| PPMQCHAR | <code>typedef PMQCHAR MQPOINTER PPMQCHAR;</code> |
| PPMQCNO | <code>typedef PMQCNO MQPOINTER PPMQCNO;</code> |
| PPMQGMO | <code>typedef PMQGMO MQPOINTER PPMQGMO;</code> |
| PPMQHCONN | <code>typedef PMQHCONN MQPOINTER PPMQHCONN;</code> |
| PPMQHOBJ | <code>typedef PMQHOBJ MQPOINTER PPMQHOBJ;</code> |
| Ppmqlong | <code>typedef PMQLONG MQPOINTER PPMQLONG;</code> |

| 数据类型 | 表示 |
|-----------|--|
| PPMQMD | <code>typedef PMQMD MQPOINTER PPMQMD;</code> |
| PPMQOD | <code>typedef PMQOD MQPOINTER PPMQOD;</code> |
| PPMQPMO | <code>typedef PMQPMO MQPOINTER PPMQPMO;</code> |
| PPMQULONG | <code>typedef PMQULONG MQPOINTER PPMQULONG;</code> |
| PPMQVOID | <code>typedef PMQVOID MQPOINTER PPMQVOID;</code> |

其中 `defined(MQ_64_BIT)` 表示 64 位平台。

请参阅第 236 页的『数据类型』以获取 MQPOINTER 宏变量的描述。

COBOL 声明

| 数据类型 | 表示 |
|----------|-------------------------------|
| MQBOOL | <code>PIC S9(9) BINARY</code> |
| MQBYTE | <code>PIC X</code> |
| MQBYTE8 | <code>PIC X(8)</code> |
| MQBYTE16 | <code>PIC X(16)</code> |
| MQBYTE24 | <code>PIC X(24)</code> |
| MQBYTE32 | <code>PIC X(32)</code> |
| MQBYTE40 | <code>PIC X(40)</code> |
| MQCHAR | <code>PIC X</code> |
| MQCHAR4 | <code>PIC X(4)</code> |
| MQCHAR8 | <code>PIC X(8)</code> |
| MQCHAR12 | <code>PIC X(12)</code> |
| MQCHAR20 | <code>PIC X(20)</code> |
| MQCHAR28 | <code>PIC X(28)</code> |

| 数据类型 | 表示 |
|-----------|-------------------|
| 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 |

PL/I 声明
PL/I 在 z/OS 上受支持。

| 数据类型 | 表示 |
|--------|---------------|
| MQBOOL | fixed bin(31) |

| 数据类型 | 表示 |
|-----------|-----------------------|
| 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) |

| 数据类型 | 表示 |
|----------|---------------|
| 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) |

System/390 汇编程序声明
System/390 汇编程序仅在 z/OS 上受支持。

| 数据类型 | 表示 |
|----------|---------|
| 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 |

| 数据类型 | 表示 |
|-----------|----------|
| 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 |

结构数据类型-简介

本部分介绍了 MQI 中使用的结构数据类型。在后续部分中描述了结构数据类型本身。

摘要

下表汇总了 MQI 中使用的结构数据类型。

| 表 467: MQI 调用 (或出口函数) 上使用的结构数据类型: | | |
|-----------------------------------|---------------|---|
| 结构 | 描述 | 使用时的调用 |
| MQACH | API 出口链头 | |
| MQAIR | 认证信息记录 | MQCONN |
| MQAXC | API 出口上下文 | |
| MQAXP | API 出口参数 | |
| MQBMHO | 缓冲区到消息句柄选项 | MQBUFMH |
| MQBO | 开始选项 | MQBEGIN |
| MQCBD | 回调描述符 | MQCB |
| MQCBO | Create-bag 选项 | mqCreate 包 |
| MQCHARV | 可变长度字符串 | MQINQMP |
| MQCNO | 连接选项 | MQCONN |
| MQCSP | 安全性参数 | MQCONN |
| MQCTLO | 回调选项 | MQCTL |
| MQDMPO | 删除消息属性选项 | MQDLTMP |
| MQGMO | 获取消息选项 | MQGet |
| MQIMPO | 查询消息属性选项 | MQINQMP |
| MQMD | 消息描述符 | MQBUFMH , MQMHBUF , MQCB , MQGET , MQPUT , MQPUT1 |
| MQMHBO | 消息句柄到缓冲区选项 | MQMHBUF |
| MQOD | 对象描述符 | MQOPEN , MQPUT1 |
| MQOR | 对象记录 | MQOPEN , MQPUT1 |
| MQPD | 属性描述符 | MQSETMP |
| MQPMO | 放置消息选项 | MQPUT , MQPUT1 |
| MQPMR | 放置消息记录 | MQPUT , MQPUT1 |
| MQRR | 响应记录 | MQOPEN , MQPUT 和 MQPUT1 |
| MQSCO | SSL 配置选项 | MQCONN |
| MQSD | 预订描述符 | MQSUB |
| MQSMPO | 设置消息属性选项 | MQSETMP |
| MQSRO | 预订请求选项 | MQSUBRQ |
| MQSTS | 状态报告结构 | MQSTAT |

表 468: 消息数据中使用的结构数据类型:

| 结构 | 描述 |
|------------------------|----------------|
| MQCIH | CICS 信息头 |
| MQCFH | PCF 头 |
| MQEPH | 嵌入式 PCF 头 |
| MQDH | 分发头 |
| MQDLH | 死信 (未传递的消息) 头 |
| MQIIH | IMS 信息头 |
| MQMDE | 消息描述符扩展 |
| MQRFH | 规则和格式化头 |
| MQRFH2 | 规则和格式化头 2 |
| MQRMH | 参考消息头 |
| MQTM | 触发器消息 |
| MQTMC2 | 触发器消息 (字符格式 2) |
| MQWIH | 工作信息头 |
| MQXQH | 传输队列头 |

注: MQDXP 结构 (数据转换出口参数) 与关联的数据转换调用一起在 [第 791 页](#) 的『数据转换出口』中描述。

结构数据类型的规则

编程语言对结构的支持程度各不相同, 采用某些规则和约定在每种编程语言中一致地映射 MQI 结构:

1. 结构必须在其自然边界上对齐。
 - 大多数 MQI 结构都需要 4 字节对齐。
 - 在 IBM i 上, 包含指针的结构需要 16 字节对齐; 这些是 :MQCNO, MQOD 和 MQPMO。
2. 结构中的每个字段都必须在其自然边界上对齐。
 - 具有等同于 MQLONG 的数据类型的字段必须在 4 字节边界上对齐。
 - 具有等同于 MQPTR 的数据类型的字段必须在 IBM i 上的 16 字节边界上对齐, 在其他环境中必须在 4 字节边界上对齐。
 - 其他字段在 1 字节边界上对齐。
3. 结构的长度必须是其边界对齐的倍数。
 - 大多数 MQI 结构的长度是 4 字节的倍数。
 - 在 IBM i 上, 包含指针的结构长度是 16 字节的倍数。
4. 必要时, 必须添加填充字节或字段以确保符合上述规则。

描述中使用的约定

每种结构数据类型的描述包括:

- 结构的用途和用途概述
- 结构中字段的描述, 格式独立于编程语言
- 如何在每种受支持的编程语言中声明结构的示例

每种结构数据类型的描述都包含以下部分:

结构名称

结构的名称，后跟结构中字段的摘要。

概述

对结构的用途和用途的简要描述。

字段

字段的描述。对于每个字段，该字段的名称后跟括号 () 中的基本数据类型。在文本中，使用斜体字面显示字段名称；例如 *Version*。

还有对字段用途的描述，以及字段可以采用的任何值的列表。常量的名称以大写形式显示；例如 MQGMO_STRUC_ID。使用 * 字符显示一组具有相同前缀的常量，例如 :MQIA_*。

在这些字段的描述中，使用了以下术语：

输入

在进行呼叫时提供字段中的信息。

输出

当调用完成或失败时，队列管理器将在字段中返回信息。

输入/输出

您在进行调用时在字段中提供信息，当调用完成或失败时，队列管理器会更改信息。

初始值

显示 MQI 随附的数据定义文件中每个字段的初始值的表。

C 声明

C 中结构的典型声明。

COBOL 声明

COBOL 中结构的典型声明。

PL/I 声明

PL/I 中结构的典型声明。

System/390 汇编程序声明

System/390 汇编语言中结构的典型声明。

Visual Basic 声明

Visual Basic 中结构的典型声明。

C 编程

本部分包含帮助您使用来自 C 编程语言的 MQI 的信息。

头文件

提供了头文件以帮助您编写使用 MQI 的 C 应用程序。

这些头文件在 [第 235 页的表 469](#) 中进行了摘要。

| 文件 | 内容 |
|-------|---|
| CMQC | 主 MQI 的函数原型，数据类型和命名常量 |
| CMQXC | 数据转换出口的函数原型，数据类型和命名常量 |
| CMQEC | 主要 MQI，数据转换出口和接口入口点结构的函数原型，数据类型和命名常量 (CMQEC 包括 CMQXC 和 CMQC。) |

要提高应用程序的可移植性，请在 `#include` 预处理器伪指令上以小写形式对头文件的名称进行编码：

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

函数

您不需要指定每次调用函数时由地址传递的所有参数。

- 按值传递 仅输入 且类型为 MQHCONN, MQHOBJ 或 MQLONG 的参数。
- 按地址传递所有其他参数。

如果不需要特定参数, 请使用空指针作为函数调用上的参数, 以代替参数数据的地址。 可以进行此操作的参数在调用描述中被识别。

不返回任何参数作为函数的值; 在 C 术语中, 这意味着所有函数都返回 void。

函数的属性由 MQENTRY 宏变量定义; 此宏变量的值取决于环境。

具有未定义数据类型的参数

MQGET, MQPUT 和 MQPUT1 函数上的 *Buffer* 参数具有未定义的数据类型。 此参数用于发送和接收应用程序的消息数据。

此类参数在 C 示例中显示为 MQBYTE 的数组。 您可以通过这种方式声明参数, 但通常更便于将它们声明为描述消息中数据布局的特定结构。 将实际函数参数声明为指向空的指针, 并将任何类型的数据的地址指定为函数调用上的参数。

数据类型

使用 C typedef 语句定义所有数据类型。 对于每种数据类型, 还定义相应的指针数据类型。 指针数据类型的名称是以字母 P (表示指针) 为前缀的基本或结构数据类型的名称。 使用 MQPOINTER 宏变量定义指针的属性; 此宏变量的值取决于环境。 下面说明了如何声明指针数据类型:

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

处理二进制字符串

将二进制数据的字符串声明为 MQBYTEn 数据类型之一。

每当复制, 比较或设置此类型的字段时, 请使用 C 函数 memcpy, memcmp 或 memset; 例如:

```
#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));
```

请勿使用字符串函数 strcpy, strcmp, strncpy 或 strncmp, 因为这些函数无法正确用于使用 MQBYTEn 数据类型声明的数据。

处理字符串

当队列管理器向应用程序返回字符串数据时, 队列管理器始终将字符串数据用空白填充到定义的字段长度; 队列管理器不返回以 null 结束的字符串。

因此, 在复制, 比较或并置此类字符串时, 请使用字符串函数 strncpy, strncmp 或 strncat。

请勿使用要求字符串由 null 终止的字符串函数 (strcpy, strcmp, strcat)。 此外, 请勿使用函数 strlen 来确定字符串的长度; 而是使用 sizeof 函数来确定字段的长度。

结构的初始值

头文件定义各种宏变量, 当您声明这些结构的实例时, 可以使用这些宏变量为 MQ 结构提供初始值。

这些宏变量具有 MQxxx_DEFAULT 形式的名称, 其中 MQxxx 表示结构的名称。 它们的使用方式如下:

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

对于某些字符字段 (例如, 大多数结构中出现的 *StrucId* 字段或 MQMD 中出现的 *Format* 字段), MQI 定义了有效的特定值。对于每个有效值, 将提供两个宏变量:

- 一个宏变量将值定义为具有长度的字符串, 但不包括隐含的空匹配项, 这正好是定义的字段的长度。例如, 对于 MQMD 中的 *Format* 字段, 提供了以下宏变量 (␣ 表示空白字符):

```
#define MQFMT_STRING "MQSTR␣␣␣"
```

将此表单与 `memcpy` 和 `memcmp` 函数配合使用。

- 另一个宏变量将值定义为字符数组; 此宏变量的名称是以 `_ARRAY` 为后缀的字符串格式的名称。例如:

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

当您声明具有与 `MQMD_DEFAULT` 宏变量提供的值不同的值的结构实例时, 请使用此表单来初始化该字段。(这并非总是必要的; 在某些环境中, 您可以在这两种情况下都使用值的字符串形式。但是, 您可以将数组表单用于声明, 因为这是与 C++ 编程语言兼容所必需的。)

动态结构的初始值

当需要可变数量的结构实例时, 通常会在使用 `calloc` 或 `malloc` 函数动态获取的主存储器中创建这些实例。要初始化此类结构中的字段, 请考虑以下方法:

1. 使用相应的 `MQxxx_DEFAULT` 宏变量声明结构的实例以初始化该结构。此实例将成为其他实例的模型:

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

可以在声明上对 `static` 或 `auto` 关键字进行编码, 以便根据需要为模型实例提供静态或动态生存期。

2. 使用 `calloc` 或 `malloc` 函数来获取结构的动态实例的存储器:

```
PMQMD Instance;
Instance = malloc(sizeof(MQMD)); /* get storage for dynamic instance */
```

3. 使用 `memcpy` 函数将模型实例复制到动态实例:

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

从 C++ 使用

对于 C++ 编程语言, 头文件包含仅在使用 C++ 编译器时包含的以下附加语句:

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

/* rest of header file */

#ifdef __cplusplus
}
#endif
```

符号约定

此信息显示如何调用函数和声明参数。

在某些情况下, 参数是大小不固定的数组。对于这些, 使用小写 `n` 来表示数字常量。当您对该参数的声明进行编码时, 请将 `n` 替换为所需的数字值。

COBOL 编程

本部分包含可帮助您使用 COBOL 编程语言中的 MQI 的信息。

副本文件

提供了各种 COPY 文件来帮助您编写使用 MQI 的 COBOL 应用程序。有两个文件包含命名常量，每个结构有两个文件。

每种结构都以两种形式提供：一种具有初始值的形式，另一种没有：

- 在 COBOL 程序的 WORKING-STORAGE SECTION 中使用具有初始值的结构；它们包含在以字母 V (表示值) 为后缀的名称的 COPY 文件中。
- 在 COBOL 程序的 LINKAGE SECTION 中使用没有初始值的结构；它们包含在 COPY 文件中，其名称以字母 L 为后缀 (对于 Linkage)。

在 [第 238 页的表 470](#) 中概述了 COPY 文件。并非所有列出的文件都在所有环境中可用。

| 文件 (具有初始值) | 文件 (不含初始值) | 内容 |
|------------|------------|---------------------|
| CMQAIRV | CMQAIRL | 认证信息记录 |
| CMQBOV | CMQ 提货单 | 开始选项结构 |
| CMQCIHV | CMQCIHL | CICS 信息头结构 |
| CMQCNV | CMQCNOL | 连接选项结构 |
| CMQDHV | CMQDHL | 分发头结构 |
| CMQDLHV | CMQDLHL | 死信头结构 |
| CMQDXPV | CMQDXPL | 数据转换出口参数结构 |
| CMQGMV | CMQGMOL | 获取消息选项结构 |
| CMQIIHV | CMQIIHL | IMS 信息头结构 |
| CMQMDV | CMQMDL | 消息描述符结构 |
| CMQMDEV | CMQMDEL | 消息描述符扩展结构 |
| CMQMD1V | CMQMD1L | 消息描述符结构版本 1 |
| CMQODV | CMQODL | 对象描述符结构 |
| CMQORV | CMQORL | 对象记录结构 |
| CMQPMV | CMQPMOL | Put 消息选项结构 |
| CMQRFHV | CMQRFHL | 规则和格式化头结构 |
| CMQRFH2V | CMQRFH2L | 规则和格式化头结构版本 2 |
| CMQRMHV | CMQRMHL | 参考消息头结构 |
| CMQRRV | CMQRRL | 响应记录结构 |
| CMQSCOV | CMQSCOL | SSL 配置选项 |
| CMQTMV | CMQTML | 触发器消息结构 |
| CMQTMCV | CMQTMCL | 触发器消息结构 (字符格式) |
| CMQTM2V | CMQTM2L | 触发器消息结构 (字符格式) 版本 2 |
| CMQWIHV | CMQWIHL | 工作信息头结构 |
| CMQXQHV | CMQXQHL | 传输队列头结构 |

表 470: COBOL 副本文件 (继续)

| 文件 (具有初始值) | 文件 (不含初始值) | 内容 |
|------------|------------|-------------|
| CMQV | - | 主 MQI 的命名常量 |
| CMQXV | - | 数据转换出口的指定常量 |
| CMQMD2V | CMQMD2L | 消息描述符结构版本 2 |

结构

在 COPY 文件中，每个结构声明都以 level-10 项开头；这使您能够通过 level-01 声明进行编码，然后使用 COPY 语句在结构声明的其余部分中进行复制来声明结构的多个实例。要引用相应的实例，请使用 IN 关键字：

```
* 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.
```

使结构在适当的边界上对齐。如果使用 COPY 语句在不是 level-01 项的项之后包含结构，请确保该结构从 level-01 项开始的相应偏移量开始。大多数 MQI 结构都需要 4 字节对齐；这方面的例外是 MQCNO，MQOD 和 MQPMO，它们需要在 IBM i 上进行 16 字节对齐。

在此部分中，将显示结构中的字段名称，但不带前缀。在 COBOL 中，字段名称以结构名称为前缀，后跟连字符。但是，如果结构名称以数字结尾，表明结构是原始结构的第二个或更高版本，那么将从前缀中省略数字。COBOL 中的字段名称以大写形式显示（尽管需要时可以使用小写或混合大小写）。例如，第 364 页的『MQMD - 消息描述符』中描述的字段 *MsgType* 在 COBOL 中变为 MQMD-MSGTYPE。

使用所有字段的初始值声明 V 后缀结构；您只需要设置那些需要与提供的初始值不同的值的字段。

指针

某些结构需要处理可能与该结构不相邻的可选数据，例如 MQOD 结构寻址的 MQOR 和 MQRR 记录。

为了处理此可选数据，结构包含使用指针数据类型声明的字段。但是，COBOL 在所有环境中都不支持指针数据类型。因此，也可以使用包含从结构开始的数据偏移量的字段来寻址可选数据。

如果要在环境之间移植应用程序，请确定指针数据类型是否在所有预期环境中都可用。如果不是，那么应用程序必须使用偏移量字段而不是指针字段来寻址可选数据。

在不支持指针的环境中，将指针字段声明为适当长度的字节字符串，初始值为全空字节字符串。如果您正在使用偏移量字段，请不要更改此初始值。

命名常量

在此部分中，常量名称显示为包含下划线字符 (_) 作为名称的一部分。在 COBOL 中，使用连字符 (-) 代替下划线。

具有字符串值的常量使用单引号作为字符串定界符 (')。在某些环境中，您可能必须指定适当的编译器选项，以使编译器接受单引号作为字符串定界符来代替双引号。

指定的常量在 COPY 文件中声明为 level-10 项。要使用常量，请显式声明 level-01 项，然后使用 COPY 语句在常量的声明中进行复制：

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

上述方法会导致常量占用程序中的存储器，即使未引用这些常量也是如此。如果在同一运行单元中的许多单独程序中包含常量，那么存在常量的多个副本，将不必要地使用主存储器。通过使用下列其中一种方法来避免此影响：

- 将 GLOBAL 子句添加到 level-01 声明:

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

这将导致仅为运行单元中的一组常量分配存储器。但是，这些常量可以由运行单元中的任何程序引用，而不仅仅是包含 level-01 声明的程序。

注: GLOBAL 子句在所有环境中都不受支持。

- 仅手动将该程序引用的那些常量复制到每个程序中。请勿使用 COPY 语句将所有常量复制到程序中。

符号约定

本节中的后一个主题显示了如何调用调用和声明参数。在某些情况下，参数是大小不固定的表或字符串。对于这些，使用小写 n 来表示数字常量。当您对该参数的声明进行编码时，请将 n 替换为所需的数字值。

System/390 汇编程序编程

本部分包含帮助您使用 System/390 汇编程序编程语言中的 MQI 的信息。

宏

提供了各种宏来帮助您编写使用 MQI 的汇编程序应用程序。

对于指定的常量有两个宏，对于每个结构都有一个宏。这些文件在 [第 240 页的表 471](#) 中进行了概述。

| 文件 | 内容 |
|----------|---------------------|
| CMQA | 主 MQI 的指定常量 (等于) |
| CMQCIHA | CICS 信息头结构 |
| CMQCNOA | 连接选项结构 |
| CMQDLHA | 死信头结构 |
| CMQDXPA | 数据转换出口参数结构 |
| CMQGMOA | 获取消息选项结构 |
| CMQIIHA | IMS 信息头结构 |
| CMQMDA | 消息描述符结构 |
| CMQMDEA | 消息描述符扩展结构 |
| CMQODA | 对象描述符结构 |
| CMQPMOA | Put 消息选项结构 |
| CMQRFHA | 规则和格式化头结构 |
| CMQRFH2A | 规则和格式化头结构版本 2 |
| CMQRMHA | 参考消息头结构 |
| CMQTMA | 触发器消息结构 |
| CMQTMC2A | 触发器消息结构 (字符格式) 版本 2 |
| CMQVERA | 结构版本控制 |
| CMQWIHA | 工作信息头结构 |
| CMQXA | 数据转换出口的指定常量 |
| CMQXPA | API 交叉出口参数结构 |

表 471: 汇编程序宏 (继续)

| 文件 | 内容 |
|---------|---------|
| CMQXQHA | 传输队列头结构 |

结构

这些结构由具有各种参数来控制宏的操作的宏生成。以下部分描述了这些参数。

不时引入新版本的 MQ 结构。新版本中的其他字段可能会导致先前小于 256 字节的结构变为大于 256 字节。因此，请编写旨在复制 MQ 结构或将 MQ 结构设置为空的汇编程序指令，以正确处理可能大于 256 个字节的结构。或者，使用 DCLVER 宏参数或带有 VERSION 参数的 CMQVERA 宏来声明结构的特定版本。

指定结构的名称

要声明一个结构的多个实例，宏以用户可指定的字符串和下划线作为结构中每个字段的名称的前缀。

使用的字符串是在调用宏时指定的标签。如果未指定标签，那么将使用结构的名称来构造前缀：

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

此部分中显示的结构声明使用缺省前缀。

指定结构的格式

结构声明可以由宏以两种格式之一生成，由 DSECT 参数控制：

DSECT=YES

汇编程序 DSECT 指令用于启动新的数据部分；结构定义紧跟在 DSECT 语句之后。宏调用中的标签用作数据段的名称；如果未指定标签，那么将使用结构的名称。

DSECT=NO

汇编程序 DC 指示信息用于在例程中的当前位置定义结构。这些字段使用值进行初始化，这些值可以通过对宏调用上的相关参数进行编码来指定。宏调用中未指定值的字段使用缺省值进行初始化。

指定的值必须为大写。如果未指定 DSECT 参数，那么将采用 DSECT=NO。

控制结构的版本

缺省情况下，宏始终声明每个结构的最新版本。

虽然可以使用 VERSION 宏参数为结构中的 *Version* 字段指定值，但该参数定义 *Version* 字段的初始值，并且不控制实际声明的结构版本。要控制声明的结构版本，请使用 DCLVER 参数：

DCLVER=CURRENT

声明的版本是当前 (最新) 版本。

DCLVER=SPECIFIED

声明的版本是由 VERSION 参数指定的版本。如果省略 VERSION 参数，那么缺省值为 V1。

如果指定 VERSION 参数，那么该值必须是自定义数字常量或所需版本的命名常量 (例如，MQCNO_VERSION_3)。如果指定其他某个值，那么将声明结构，就像指定了 DCLVER=CURRENT 一样，即使 VERSION 的值解析为有效值也是如此。

指定的值必须为大写。如果省略 DCLVER 参数，那么将从 MQDCLVER 全局宏变量中获取所使用的值。可以使用 CMQVERA 宏设置此变量。

声明嵌入在另一个结构中的一个结构

要将一个结构声明为另一个结构的组件，请使用 NESTED 参数：

NESTED=YES

结构声明嵌套在另一个结构声明中。

NESTED=NO

结构声明未嵌套在另一个结构声明中。

指定的值必须为大写。如果省略 NESTED 参数，那么将采用 NESTED=NO。

指定字段的初始值

指定要用于在结构中初始化字段的值，方法是将该字段的名称 (不带前缀) 编码为宏调用上的参数，并附带所需的值。

例如，要声明使用 MQMT_REQUEST 初始化的 *MsgType* 字段和使用字符串 "MY_REPLY_TO_QUEUE" 初始化的 *ReplyToQ* 字段的消息描述符结构，请使用以下命令：

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

如果指定命名常量 (等于) 作为宏调用上的值，请使用 CMQA 宏来定义命名常量。请勿将字符串值括在单引号中。

控制列表

使用 LIST 参数控制汇编程序列表中结构声明的外观：

LIST=YES

结构声明出现在汇编程序列表中。

LIST=NO

结构声明不会出现在汇编程序列表中。

指定的值必须为大写。如果省略 LIST 参数，那么将采用 LIST=NO。

CMQVERA 宏

此宏允许您设置要用于结构宏上的 DCLVER 参数的缺省值。仅当从结构宏的调用中省略 DCLVER 参数时，CMQVERA 指定的值才由结构宏使用。通过使用 DCLVER 参数对 CMQVERA 宏进行编码来设置缺省值：

DCLVER=CURRENT

缺省版本设置为当前 (最新) 版本。

DCLVER=SPECIFIED

缺省版本设置为 VERSION 参数指定的版本。

必须指定 DCLVER 参数，并且该值必须为大写。在下次调用 CMQVERA 或组合件结束之前，CMQVERA 设置的值将保留为缺省值。如果省略 CMQVERA，那么缺省值为 DCLVER=CURRENT。

符号约定

后面的部分显示了如何调用调用和声明参数。在某些情况下，参数是大小不固定的数组或字符串，使用小写 n 来表示数字常量。当您对该参数的声明进行编码时，请将 n 替换为所需的数字值。

MQAIR-认证信息记录

MQAIR 结构表示认证信息记录。

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|------------------|---------------------------|----------------------------------|
| StrucId | 结构标识 | StrucId |
| 版本 | 结构版本号 | 版本 |
| AuthInfoType | 认证信息的类型 | AuthInfoType |
| AuthInfoConnName | LDAP CRL 服务器的连接名称 | AuthInfoConnName |
| LDAPUserNamePtr | LDAP 用户名的地址 | LDAPUserNamePtr |
| LDAPUserName 偏移量 | LDAP 用户名与 MQSCO 开始处之间的偏移量 | LDAPUserName 偏移量 |

表 472: MQAIR 中的字段 (继续)

| 字段 | 描述 | Topic |
|---------------------------------------|---------------------|----------------------------------|
| LDAPUserName 长度 | LDAP 用户名的长度 | LDAPUserName 长度 |
| LDAPPASSWORD | 用于访问 LDAP 服务器的密码 | LDAPPASSWORD |
| 注: 如果版本小于 MQAIR_VERSION_2, 那么将忽略其余字段。 | | |
| OCSPResponderURL | 可与 OCSP 响应程序联系的 URL | OCSPResponderURL |

MQAIR 概述

MQAIR 结构允许作为 WebSphere MQ MQI 客户机运行的应用程序指定有关要用于客户机连接的认证程序的信息。该结构是 MQCONN 调用上的输入参数。

可用性: AIX, HP-UX, Solaris, Linux 和 Windows 客户机。

字符集和编码: MQAIR 中的数据必须采用本地队列管理器的字符集和编码; 这些数据由 **CodedCharSetId** 队列管理器属性和 MQENC_NATIVE 提供。

MQAIR 的字段

MQAIR 结构包含以下字段; 这些字段以 **字母顺序** 进行描述:

AuthInfoConnName (MQCHAR264)

这是运行 LDAP 服务器的主机的主机名或网络地址。可以后跟一个可选端口号, 用括号括起。缺省端口号为 389。

如果该值比字段的长度短, 那么用空字符终止该值, 或用空白填充该字段的长度。如果该值无效, 那么调用将失败, 原因码为 MQRC_AUTH_INFO_CONN_NAME_ERROR。

这是一个输入字段。此字段的长度由 MQ_AUTH_INFO_CONN_NAME_LENGTH 给出。此字段的初始值是 C 中的空字符串, 以及其他编程语言中的空白字符。

AuthInfo 类型 (MQLONG)

这是记录中包含的认证信息的类型。

该值可以是以下两个参数之一:

MQAIT_CRL_LDAP

使用 LDAP 服务器进行证书撤销检查。

MQAIT_OCSP

使用 OCSP 进行证书撤销检查。

如果该值无效, 那么调用将失败, 原因码为 MQRC_AUTH_INFO_TYPE_ERROR。

这是一个输入字段。此字段的初始值为 MQAIT_CRL_LDAP。

LDAPPASSWORD (MQCHAR32)

这是访问 LDAP CRL 服务器所需的密码。如果该值比字段的长度短, 那么用空字符终止该值, 或用空白填充该字段的长度。

如果 LDAP 服务器不需要密码, 或者您省略了 LDAP 用户名, 那么 *LDAPPASSWORD* 必须为空或空白。如果省略 LDAP 用户名, 并且 *LDAPPASSWORD* 不为空, 那么调用将失败, 原因码为 MQRC_LDAP_PASSWORD_ERROR。

这是一个输入字段。此字段的长度由 MQ_LDAP_PASSWORD_LENGTH 给出。此字段的初始值是 C 中的空字符串, 以及其他编程语言中的空白字符。

LDAPUserName 长度 (MQLONG)

这是 *LDAPUserNamePtr* 或 *LDAPUserNameOffset* 字段所寻址的 LDAP 用户名的长度 (以字节计)。该值必须在 0 到 MQ_ 专有名称长度的范围内。如果该值无效, 那么调用将失败, 原因码为 MQRC_LDAP_USER_NAME_LENGTH_ERR。

如果涉及的 LDAP 服务器不需要用户名, 请将此字段设置为零。

这是一个输入字段。此字段的初始值为 0。

LDAPUserName 偏移量 (MQLONG)

这是从 MQAIR 结构开始的 LDAP 用户名的偏移量 (以字节为单位)。

偏移可以是正数或负数。如果 *LDAPUserNameLength* 为零, 那么将忽略该字段。

您可以使用 *LDAPUserNamePtr* 或 *LDAPUserNameOffset* 来指定 LDAP 用户名, 但不能同时指定两者; 请参阅 *LDAPUserNamePtr* 字段的描述以获取详细信息。

这是一个输入字段。此字段的初始值为 0。

LDAPUserNamePtr (PMQCHAR)

这是 LDAP 用户名。

它由尝试访问 LDAP CRL 服务器的用户的专有名称组成。如果该值短于 *LDAPUserNameLength* 指定的长度, 请使用空字符终止该值, 或者用空格填充该值至长度 *LDAPUserNameLength*。如果 *LDAPUserNameLength* 为零, 那么将忽略该字段。

您可以通过以下两种方法之一来提供 LDAP 用户名:

- 通过使用指针字段 *LDAPUserNamePtr*

在这种情况下, 应用程序可以声明与 MQAIR 结构分开的字符串, 并将 *LDAPUserNamePtr* 设置为该字符串的地址。

请考虑将 *LDAPUserNamePtr* 用于以可移植到不同环境 (例如 C 编程语言) 的方式支持指针数据类型的编程语言。

- 通过使用偏移量字段 *LDAPUserNameOffset*

在这种情况下, 应用程序必须声明包含 MQSCO 结构的复合结构, 后跟 MQAIR 记录数组, 后跟 LDAP 用户名字符串, 并将 *LDAPUserNameOffset* 设置为相应名称字符串从 MQAIR 结构开始的偏移量。确保此值正确, 并且具有可在 MQLONG 中容纳的值 (最严格的编程语言是 COBOL, 其有效范围为 -999 999 999 到 +999 999 999 999)。

请考虑将 *LDAPUserNameOffset* 用于不支持指针数据类型的编程语言, 或者以无法移植到不同环境 (例如, COBOL 编程语言) 的方式实现指针数据类型的编程语言。

无论选择哪种技术, 都只能使用 *LDAPUserNamePtr* 和 *LDAPUserNameOffset* 之一; 如果两者都非零, 那么调用将失败, 原因码为 MQRC_LDAP_USER_NAME_ERROR。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针, 否则为全空字节字符串。

注: 在编程语言不支持指针数据类型的平台上, 此字段声明为适当长度的字节字符串。

OCSPResponderURL (MQCHAR256)

对于表示 OCSP 响应程序的连接详细信息的 MQAIR 结构, 此字段包含可与响应程序联系的 URL。

此字段的值是 HTTP URL。此字段优先于 AuthorityInfo 访问 (AIA) 证书扩展中的 URL。

除非以下两个语句都为 true, 否则将忽略该值:

- MQAIR 结构为 V 2 或更高版本 ("版本" 字段设置为 MQAIR_VERSION_2 或更高版本)。
- AuthInfo 类型字段设置为 MQAIT_OCSP。

如果该字段未包含正确格式的 HTTP URL (并且未被忽略), 那么 MQCONN 调用将失败, 原因码为 MQRC_OCSP_URL_ERROR。

此字段区分大小写。它必须以小写的字符串 http:// 开头。其余 URL 可能区分大小写, 具体取决于 OCSP 服务器实现。

此字段不受数据转换限制。

StrucId (MQCHAR4)

该值必须为:

MQAIR_STRUC_ID

认证信息记录的标识。

对于 C 编程语言，还定义了常量 MQAIR_STRUC_ID_ARRAY; 这与 MQAIR_STRUC_ID 具有相同的值，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQAIR_STRUC_ID。

Version (MQLONG)

MQAIR 结构的版本号。

值必须为以下其中一项:

MQAIR_VERSION_1

Version-1 认证信息记录。

MQAIR_VERSION_2

Version-2 认证信息记录。

以下常量指定当前版本的版本号:

MQAIR_CURRENT_VERSION

当前版本的认证信息记录。

这始终是一个输入字段。此字段的初始值为 MQAIR_VERSION_1。

MQAIR 的初始值和语言声明

| 表 473: MQAIR 中字段的初始值 | | |
|----------------------|-----------------|---------|
| 字段名称 | 常量的名称 | 常量值 |
| StrucId | MQAIR_STRUC_ID | 'AIR↵' |
| 版本 | MQAIR_VERSION_1 | 1 |
| AuthInfoType | MQAIT_CRL_LDAP | 1 |
| AuthInfoConnName | None | 空字符串或空白 |
| LDAPUserNamePtr | None | 空指针或空字节 |
| LDAPUserName 偏移量 | None | 0 |
| LDAPUserName 长度 | None | 0 |
| LDAPPassword | None | 空字符串或空白 |
| OCSResponderURL | None | 空字符串或空白 |

注意:

1. 符号 ↵ 表示单个空白字符。
2. 在 C 编程语言中，宏变量 MQAIR_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQAIR MyAIR = {MQAIR_DEFAULT};
```

C 声明

```
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 声明

```

** 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).

```

Visual Basic 声明

```

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-缓冲区到消息句柄选项

下表汇总了结构中的字段。MQBMHO 结构-缓冲区到消息句柄选项

| 表 474: MQBMHO 中的字段 | | |
|--------------------|-----------------|-------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 控制 MQBMHO 操作的选项 | 选项 |

MQBMHO 概述

可用性: 全部。缓冲区到消息句柄选项结构-概述

用途:MQBMHO 结构允许应用程序指定用于控制如何从缓冲区生成消息句柄的选项。此结构是 MQBUFMH 调用上的输入参数。

字符集和编码:MQBMHO 中的数据必须位于应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQBMHO 的字段

缓冲区到消息句柄选项结构-字段

MQBMHO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

Options (MQLONG)

缓冲区到消息句柄结构-"选项" 字段

值可以是:

MQBMHO_DELETE_PROPERTIES

将从缓冲区中删除添加到消息句柄的属性。如果调用失败, 那么不会删除任何属性。

缺省选项: 如果不需要所描述的选项, 请使用以下选项:

MQBMHO_NONE

未指定任何选项。

这始终是一个输入字段。此字段的初始值为 MQBMHO_DELETE_PROPERTIES。

StrucId (MQCHAR4)

缓冲区到消息句柄结构- StrucId 字段

这是结构标识。该值必须为:

MQBMHO_STRUC_ID

缓冲区到消息句柄结构的标识。

对于 C 编程语言, 还定义了常量 MQBMHO_STRUC_ID_ARRAY; 此值与 MQBMHO_STRUC_ID 相同, 但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQBMHO_STRUC_ID。

Version (MQLONG)

缓冲区到消息句柄结构-版本字段

这是结构版本号。该值必须为:

MQBMHO_VERSION_1

缓冲区到消息句柄结构的版本号。

以下常量指定当前版本的版本号:

MQBMHO_CURRENT_VERSION

当前版本的缓冲区到消息句柄结构。

这始终是一个输入字段。此字段的初始值为 MQBMHO_VERSION_1。

MQBMHO 的初始值和语言声明

缓冲区到消息句柄结构-初始值

| 字段名称 | 常量的名称 | 常量值 |
|----------------|------------------|--------|
| <i>StrucId</i> | MQBMHO_STRUC_ID | 'BMHO' |
| <i>Version</i> | MQBMHO_VERSION_1 | 1 |
| <i>Options</i> | MQBMHO_NONE | 0 |

表 475: MQBMHO 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|--|-------|-----|
| 注意: | | |
| 1. 在 C 编程语言中, 宏变量 MQBMHO_DEFAULT 包含以上列出的值。 通过以下方式使用它来为结构中的字段提供初始值: | | |
| <pre>MQBMHO MyBMHO = {MQBMHO_DEFAULT};</pre> | | |

C 声明

缓冲区到消息句柄结构-C 语言声明

```
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 声明

缓冲区到消息句柄结构-COBOL 语言声明

```
** 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.
```

PL/I 声明

缓冲区到消息句柄结构-PL/I 语言声明

```
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 */
```

High Level Assembler 声明

缓冲区到消息句柄结构-汇编程序语言声明

```
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-开始选项

下表汇总了结构中的字段。

| 表 476: MQBO 中的字段 | | |
|------------------|---------------------|-------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 用于控制 MQBEGIN 的操作的选项 | 选项 |

MQBO 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows; 不可用于 WebSphere MQ MQI 客户机。

用途:MQBO 结构允许应用程序指定与创建工作单元相关的选项。该结构是 MQBEGIN 调用上的输入/输出参数。

字符集和编码:MQBO 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 MQ MQI 客户机运行, 那么该结构必须采用客户机的字符集和编码。

MQBO 的字段

MQBO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

选项 (MQLONG)

此字段始终是输入字段。其初始值为 MQBO_NONE。

该值必须为:

MQBO_NONE

未指定任何选项。

StrucId (MQCHAR4)

此字段始终是输入字段。其初始值为 MQBO_STRUC_ID。

该值必须为:

MQBO_STRUC_ID

开始选项结构的标识。

对于 C 编程语言, 还定义了常量 MQBO_STRUC_ID_ARRAY; 此值与 MQBO_STRUC_ID 相同, 但是字符数组而不是字符串。

Version (MQLONG)

此字段始终是输入字段。其初始值为 MQBO_VERSION_1。

该值必须为:

MQBO_VERSION_1

开始选项结构的版本号。

以下常量指定当前版本的版本号:

MQBO_CURRENT_VERSION

当前版本的开始选项结构。

MQBO 的初始值和语言声明

| 表 477: MQBO for MQBO 中字段的初始值 | | |
|------------------------------|----------------|-------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQBO_STRUC_ID | 'BO-' |
| <i>Version</i> | MQBO_VERSION_1 | 1 |

表 477: MQBO for MQBO 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|---------|-----------|-----|
| Options | MQBO_NONE | 0 |

注意:

1. 符号 `\n` 表示单个空白字符。
2. 在 C 编程语言中, 宏变量 `MQBO_DEFAULT` 包含上面列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQBO MyBO = {MQBO_DEFAULT};
```

C 声明

```
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 声明

```
** 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.
```

PL/I 声明

```
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 */
```

Visual Basic 声明

```
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-回调上下文

下表汇总了结构中的字段。描述回调例程的结构。

表 478: MQCBC 中的字段

| 字段 | 描述 | Topic |
|---------|------|-------------------------|
| StrucID | 结构标识 | StrucID |

表 478: MQCBC 中的字段 (继续)

| 字段 | 描述 | Topic |
|--------------------------------------|-----------------|--------------------------------|
| <i>Version</i> | 结构版本号 | 版本 |
| <i>CallType</i> | 为何调用函数 | CallType |
| <i>Hobj</i> | 对象句柄 | Hobj |
| <i>CallbackArea</i> | 要使用的回调函数的字段 | CallbackArea |
| <i>ConnectionArea</i> | 要使用的回调函数的字段 | ConnectionArea |
| <i>CompCode</i> | 完成代码 | CompCode |
| <i>Reason</i> | 原因码 | 原因 |
| <i>State</i> | 指示当前使用者的状态 | 状态 |
| <i>DataLength</i> | 消息长度 | DataLength |
| <i>BufferLength</i> | 消息缓冲区的长度 (以字节计) | BufferLength |
| <i>Flags</i> | 常规标志 | 标志 |
| 注: 如果版本小于 MQCBC_VERSION_2, 那么将忽略其余字段 | | |
| <i>ReconnectDelay</i> | 重新连接尝试之前的毫秒数 | ReconnectDelay |

MQCBC 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: MQCBC 结构用于指定传递给回调函数的上下文信息。

该结构是对消息使用者例程的调用上的输入/输出参数。

版本: MQCBC 的当前版本为 MQCBC_VERSION_2。

字符集和编码: MQCBC 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 MQ MQI 客户机运行, 那么结构将采用客户机的字符集和编码。

MQCBC 的字段

MQCBC 结构的字段的字母列表。

MQCBC 结构包含以下字段; 这些字段按字母顺序描述:

BufferLength (MQLONG)

此字段是传递到此函数的消息缓冲区的长度 (以字节计)。

缓冲区可以大于为使用者定义的 MaxMsg 长度值和 MQGMO 中的 ReturnedLength 值。

实际消息长度在 [DataLength](#) 字段中提供。

在回调函数的持续时间内, 应用程序可以将整个缓冲区用于自己的目的。

这是消息使用者函数的输入字段; 它与异常处理程序函数无关。

CallbackArea (MQPTR)

此字段可供回调函数使用。

队列管理器不会根据此字段的内容做出任何决策, 并且会从 MQCBD 结构中的第 258 页的『[CallbackArea](#) (MQPTR)』字段 (这是用于定义回调函数的 MQCB 调用上的参数) 未更改地传递此决策。

在 *HObj* 的回调函数的调用中保留对 *CallbackArea* 的更改。此字段不与其他句柄的回调函数共享。这是回调函数的输入/输出字段。此字段的初始值为空指针或空字节。

CallType (MQLONG)

包含有关调用此函数的原因的信息的字段; 定义了以下内容。

消息传递调用类型: 这些调用类型包含有关消息的信息。 *DataLength* 和 *BufferLength* 参数对这些调用类型有效。

MQCBCT_MSG_REMOVED

已使用已从对象句柄以破坏性方式除去的消息来调用消息使用者函数。

如果 *CompCode* 的值为 MQCC_WARNING, 那么 *Reason* 字段的值为 MQRC_TRUNCATED_MSG_RECEIVED 或指示数据转换问题的其中一个代码。

MQCBCT_MSG_NOT_REMOVED

已使用尚未以破坏性方式从对象句柄中除去的消息来调用消息使用者函数。可以使用 *MsgToken* 以破坏性方式从对象句柄中除去消息。

可能未除去该消息, 因为:

- MQGMO 选项请求了浏览操作 MQGMO_BROWSE_*
- 消息大于可用缓冲区, 并且 MQGMO 选项未指定 MQGMO_ACCEPT_TRUNCATED_MSG

如果 *CompCode* 的值为 MQCC_WARNING, 那么 *Reason* 字段的值为 MQRC_TRUNCATED_MSG_FAILED 或指示数据转换问题的其中一个代码。

回调控制调用类型: 这些调用类型包含有关回调控制的信息, 并且不包含有关消息的详细信息。这些调用类型是使用 MQCBD 结构中的 选项 请求的。

DataLength 和 *BufferLength* 参数对于这些调用类型无效。

MQCBCT_REGISTER_CALL

此调用类型的目的是允许回调函数执行一些初始设置。

在注册回调之后, 即使用 MQOP_REGISTER 的 *Operation* 字段的值从 MQCB 调用返回时, 将立即调用回调函数。

此调用类型同时用于消息使用者和事件处理程序。

如果请求, 这是回调函数的第一次调用。

Reason 字段的值为 MQRC_NONE。

MQCBCT_START_CALL

此调用类型的目的是允许回调函数在启动时执行某些设置, 例如, 恢复先前停止时清除的资源。

使用 MQOP_START 或 MQOP_START_WAIT 启动连接时, 将调用回调函数。

如果在另一个回调函数中注册了回调函数, 那么在回调返回时将调用此调用类型。

此调用类型仅用于消息使用者。

Reason 字段的值为 MQRC_NONE。

MQCBCT_STOP_CALL

此调用类型的目的是允许回调函数在停止一段时间时执行一些清除, 例如, 清除在使用消息期间获取的其他资源。

当使用 MQOP_STOP 的 *Operation* 字段值发出 MQCTL 调用时, 将调用回调函数。

此调用类型仅用于消息使用者。

Reason 字段的值设置为指示停止的原因。

MQCBCT_DEREGISTER_CALL

此调用类型的目的是允许回调函数在使用过程结束时执行最终清除。在下列情况下调用回调函数:

- 使用带有 MQOP_DEREGISTER 的 MQCB 调用来注销回调函数。
- 队列已关闭，导致隐式注销。在此实例中，回调函数将作为对象句柄传递 MQHO_UNUSABLE_HOBJ。
- MQDISC 调用完成-导致隐式关闭，并因此导致注销。在这种情况下，不会立即断开连接，并且尚未落实任何正在进行的事务。

如果在回调函数本身内执行其中任何操作，那么一旦回调返回，将调用该操作。

此调用类型同时用于消息使用者和事件处理程序。

如果请求，那么这是回调函数的最后一次调用。

Reason 字段的值设置为指示停止的原因。

MQCBCT_EVENT_CALL

事件处理程序函数

当队列管理器或连接停止或停顿时，已在没有消息的情况下调用事件处理程序函数。

此调用可用于对所有回调函数执行相应的操作。

消息使用者函数

当检测到特定于对象句柄的错误 (*CompCode* = MQCC_FAILED) 时，调用了消息使用者函数而没有消息；例如，*Reason* 代码 = MQRC_GET_处禁止。

Reason 字段的值设置为指示调用原因。

MQCBCT_MC_EVENT_CALL

已针对多点广播事件调用事件处理程序函数；将向事件处理程序发送 WebSphere MQ 多点广播事件，而不是 "正常" WebSphere MQ 事件。

有关 MQCBCT_MC_EVENT_CALL 的更多信息，请参阅 [多点广播异常报告](#)。

CompCode (MQLONG)

此字段是完成代码。它指示使用消息时是否有任何问题。

该值为下列其中之一：

MQCC_OK

成功完成

MQCC_WARNING

警告 (部分完成)

MQCC_FAILED

通话失败

这是一个输入字段。此字段的初始值为 MQCC_OK。

ConnectionArea (MQPTR)

此字段可供回调函数使用。

队列管理器不会根据此字段的内容做出任何决策，并且会从 MQCTLO 结构中的 [第 300 页的『ConnectionArea \(MQPTR\)』](#) 字段 (这是用于控制回调函数的 MQCTL 调用上的参数) 未更改地传递此决策。

回调函数对此字段进行的任何更改都将在回调函数的调用中保留。此区域可用于传递要由所有回调函数共享的信息。与 *CallbackArea* 不同，此区域在连接句柄的所有回调中是公共的。

这是输入和输出字段。此字段的初始值为空指针或空字节。

DataLength (MQLONG)

这是消息中应用程序数据的长度 (以字节计)。如果该值为零，那么表示消息不包含任何应用程序数据。

DataLength 字段包含消息的长度，但不一定包含传递给使用者的消息数据的长度。可能是消息已截断。使用 MQGMO 中的 ReturnedLength 字段来确定实际传递给使用者的数据量。

如果原因码指示消息已被截断，那么可以使用 DataLength 字段来确定实际消息的大小。这允许您确定容纳消息数据所需的缓冲区大小，然后发出 MQCB 调用以使用适当的值更新 MaxMsgLength。

如果指定了 MQGMO_CONVERT 选项，那么转换后的消息可能大于针对 DataLength 返回的值。在此类情况下，应用程序可能需要发出 MQCB 调用以将 MaxMsgLength 更新为大于队列管理器针对 DataLength 返回的值。

为避免消息截断问题，请将 MaxMsg 长度指定为 MQCBD_FULL_MSG_LENGTH。这将导致队列管理器在数据转换后为完整消息长度分配缓冲区。但是，请注意，即使指定了此选项，仍然可能没有足够的存储空间来正确处理请求。应用程序应始终检查返回的原因码。例如，如果无法分配足够的存储空间来转换消息，那么消息将返回到未转换的应用程序。

这是消息使用者函数的输入字段；它与事件处理程序函数无关。

Flags (MQLONG)

包含有关此使用者的信息的标志。

定义了以下选项：

MQCBCF_READA_BUFFER_EMPTY

如果使用 MQCO_QUIESCE 选项的先前 MQCLOSE 调用失败，并且原因码为 MQRC_READ_AHEAD_MSGS，那么会返回此标志。

此代码指示正在返回最后一条预读消息，并且缓冲区现在为空。如果应用程序使用 MQCO_QUIESCE) 选项发出另一个 MQCLOSE 调用，那么它将成功。

请注意，不保证为应用程序提供具有此标志集的消息，因为预读缓冲区中可能仍存在与当前选择标准不匹配的消息。在此实例中，将使用原因码 MQRC_HOBJ_QUIESCED 来调用使用者函数。

如果预读缓冲区完全为空，那么将使用 MQCBCF_READA_BUFFER_EMPTY 标志和原因码 MQRC_HOBJ_QUIESCED_NO_MSGS 来调用使用者。

这是消息使用者函数的输入字段；它与事件处理程序函数无关。

Hobj (MQHOBJ)

这是对消息使用者的调用的对象句柄。

对于事件处理程序，此值为 MQHO_NONE

如果未从队列中除去消息，那么应用程序可以使用此句柄和 "获取消息选项" 块中的消息令牌来获取消息。

这始终是一个输入字段。此字段的初始值为 MQHO_UNUSABLE_HOBJ

原因 (MQLONG)

这是限定 CompCode 的原因码。

这是一个输入字段。此字段的初始值为 MQRC_NONE。

状态 (MQLONG)

指示当前使用者的状态。当将非零原因码传递到使用者函数时，此字段对应用程序具有大多数值。

您可以使用此字段来简化应用程序编程，因为您不需要对每个原因码的行为进行编码。

这是一个输入字段。此字段的初始值为 MQCS_NONE

| 状态 | 队列管理器操作 | 常量值 |
|----------------------------------|-----------|-----|
| MQCS_NONE 此原因码表示没有其他原因信息的正常调用 | 无；这是正常操作。 | 0 |

| 状态 | 队列管理器操作 | 常量值 |
|---|--|-----|
| <i>MQCS_SUSPENDED_TEMPORARY</i> 这些原因码表示临时条件。 | 调用回调例程以报告条件，然后将其暂挂。经过一段时间后，系统可能会再次尝试该操作，这可能会导致再次发生相同的情况。 | 1 |
| <i>MQCS_SUSPENDED_USER_ACTION</i> 这些原因码表示回调需要执行操作以解析条件的条件。 | 将暂挂使用者，并调用回调例程以报告该情况。如果可能，回调例程应解析条件，并恢复或关闭连接。 | 2 |
| <i>MQCS_SUSPENDED</i> 这些原因码表示阻止进一步消息回调的故障。 | 队列管理器自动暂挂回调函数。如果恢复回调函数，那么可能再次接收到相同的原因码。 | 3 |
| <i>MQCS_STOPPED</i> 这些原因码表示消息使用结束。 | 传递到异常处理程序以及指定 <i>MQCBDO_STOP_CALL</i> 的回调。无法使用更多消息。 | 4 |

StrucId (MQCHAR4)

此字段中的值是结构标识。

该值必须为:

MQCBC_STRUC_ID

回调上下文结构的标识。

对于 C 编程语言，还定义了常量 *MQCBC_STRUC_ID_ARRAY*; 此值与 *MQCBC_STRUC_ID* 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 *MQCBC_STRUC_ID*。

Version (MQLONG)

此字段中的值是结构版本号。

该值必须为:

MQCBC_VERSION_1

Version-1 回调上下文结构。

以下常量指定当前版本的版本号:

MQCBC_CURRENT_VERSION

回调上下文结构的当前版本。

这始终是一个输入字段。此字段的初始值为 *MQCBC_VERSION_1*。

回调函数始终传递最新版本的结构。

ReconnectDelay (MQLONG)

ReconnectDelay 指示队列管理器在尝试重新连接之前将等待的时间长度。该字段可由事件处理程序修改以更改延迟或完全停止重新连接。

仅当回调上下文中 *Reason* 字段的值为 *MQRC_RECONNECTING* 时，才使用 *ReconnectDelay* 字段。

在进入事件处理程序时，*ReconnectDelay* 的值是队列管理器在进行重新连接尝试之前要等待的毫秒数。第 255 页的表 479 列出您可以设置的值，以便在从事件处理程序返回时修改队列管理器的行为。

| 名称 | 值 | 描述 |
|--------------------------|----|----------------------|
| <i>MQRD_NO_RECONNECT</i> | -1 | 不再尝试重新连接。将向应用程序返回错误。 |
| <i>MQRD_NO_DELAY</i> | 0 | 请立即尝试重新连接。 |

表 479: ReconnectDelay 值 (继续)

| 名称 | 值 | 描述 |
|--------------|----|-------------------------|
| Milliseconds | >0 | 在重试连接之前, 请等待此时间 (以毫秒计)。 |

MQCBC 的初始值和语言声明

回调上下文结构-初始值

MQCBC 结构没有初始值。此结构作为参数传递到回调例程。队列管理器初始化结构; 应用程序从不初始化该结构。

C 声明

回调上下文结构-C 语言声明

```
typedef struct tagMQCBC MQCBC;
struct tagMQCBC {
    MQCHAR4   StructId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    CallType;         /* Why Function was called */
    MQHOBJS   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 声明

```
** 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-DATALNGTH 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 **
```

PL/I 声明

```
dc1
```



```

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 */

```

High Level Assembler 声明

```

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-回调描述符

下表汇总了结构中的字段。指定回调函数的结构。

| 表 480: MQCBD 中的字段 | | |
|-------------------------|---------------------|----------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucID</i> | 结构标识 | StrucID |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>CallbackType</i> | 回调函数的类型 | CallbackType |
| <i>Options</i> | 控制消息使用的选项 | 选项 |
| <i>Callback Area</i> | 要使用的回调函数的字段 | CallbackArea |
| <i>CallbackFunction</i> | 是否将函数作为 API 调用进行调用 | CallbackFunction |
| <i>CallbackName</i> | 是否将该函数作为动态链接的程序进行调用 | CallbackName |
| <i>MaxMsgLength</i> | 可读取的最长消息的长度 | MaxMsgLength |

MQCBD 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: MQCBD 结构用于指定回调函数以及由队列管理器控制其使用的选项。

该结构是 MQCB 调用上的输入参数。

版本: MQCBD 的当前版本为 MQCBD_VERSION_1。

字符集和编码:MQCBD 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 MQ MQI 客户机运行, 那么该结构必须采用客户机的字符集和编码。

MQCBD 的字段

MQCBD 结构的字段的字母列表。

MQCBD 结构包含以下字段; 这些字段按字母顺序描述:

CallbackArea (MQPTR)

回调描述符结构- *CallbackArea* 字段

这是可供回调函数使用的字段。

队列管理器不会根据此字段的内容做出任何决策, 并且会从 MQCBC 结构中的 [第 251 页](#) 的『*CallbackArea* (MQPTR)』字段 (回调函数声明上的参数) 未更改地传递此决策。

该值仅在具有值 MQOP_REGISTER 的 *Operation* 上使用, 没有当前定义的回调, 它不会替换先前的定义。这是回调函数的输入和输出字段。此字段的初始值为空指针或空字节。

CallbackFunction (MQPTR)

回调描述符结构- *CallbackFunction* 字段

回调函数作为函数调用进行调用。

使用此字段来指定指向回调函数的指针。

必须指定 *CallbackFunction* 或 *CallbackName*。如果同时指定这两者, 那么将返回原因码 MQRC_CALLBACK_ROUTINE_ERROR。

如果既未设置 *CallbackName* 也未设置 *CallbackFunction*, 那么调用将失败, 原因码为 MQRC_CALLBACK_ROUTINE_ERROR。

此选项在以下环境中不受支持: 不支持函数指针引用的编程语言和编译器。在此类情况下, 调用失败, 原因码为 MQRC_CALLBACK_ROUTINE_ERROR。

在 z/OS 上, 必须使用操作系统链接约定来调用该函数。例如, 在 C 编程语言中, 指定:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

这是一个输入字段。此字段的初始值为空指针或空字节。

注: 将 CICS 与 WebSphere MQ V7.0.1 配合使用时, 在以下情况下支持异步使用:

- Apar PK66866 应用于 CICS TS 3.2
- Apar PK89844 应用于 CICS TS 4.1

CallbackName (MQCHAR128)

回调描述符结构- *CallbackName* 字段

回调函数作为动态链接的程序调用。

必须指定 *CallbackFunction* 或 *CallbackName*。如果同时指定这两者, 那么将返回原因码 MQRC_CALLBACK_ROUTINE_ERROR。

如果既未设置 *CallbackName* 也未设置 *CallbackFunction*, 那么调用将失败, 原因码为 MQRC_CALLBACK_ROUTINE_ERROR。

当注册要使用的第一个回调例程时装入该模块, 当最后一个要使用该模块的回调例程注销时卸载该模块。

除非在以下文本中注明, 否则名称在字段中是左对齐的, 没有嵌入的空格; 名称本身用空格填充字段的长度。在下面的描述中, 方括号 ([]) 表示可选信息:

IBM i

回调名称可以是下列其中一种格式:

- 库 "/" 程序
- 库 "/" ServiceProgram "("FunctionName")"

例如, MyLibrary/MyProgram(MyFunction)。

库名可以是 *LIBL。库名和程序名都限制为最多 10 个字符。

UNIX 系统

回调名称是动态可装入模块或库的名称, 以驻留在该库中的函数的名称作为后缀。函数名必须括在括号内。可以选择以目录路径作为库名的前缀:

```
[path]library(function)
```

如果未指定路径, 那么将使用系统搜索路径。

名称限制为最多 128 个字符。

Windows

回调名称是动态链接库的名称, 后缀为驻留在该库中的函数的名称。函数名必须用括号括起。库名可以选择以目录路径和驱动器作为前缀:

```
[d:][path]library(function)
```

如果未指定磁带机和路径, 那么将使用系统搜索路径。

名称限制为最多 128 个字符。

z/OS

回调名称是对 LINK 或 LOAD 宏的 EP 参数规范有效的装入模块的名称。

名称限制为最多 8 个字符。

z/OS CICS

回调名称是对 EXEC CICS LINK 命令宏的 PROGRAM 参数规范有效的装入模块的名称。

名称限制为最多 8 个字符。

可以使用已安装的 PROGRAM 定义的 REMOTESYTEM 选项或通过动态路由程序将程序定义为远程程序。

如果程序要使用 WebSphere MQ API 调用, 那么远程 CICS 区域必须连接到 WebSphere MQ。但是, 请注意, MQCBC 结构中的 [第 254 页的『Hobj \(MQHOBJ\)』](#) 字段在远程系统中无效。

如果尝试装入 *CallbackName* 时发生故障, 那么会将下列其中一个错误代码返回到应用程序:

- MQRC_MODULE_NOT_FOUND
- MQRC_MODULE_INVALID
- MQRC_MODULE_ENTRY_NOT_FOUND

还会将一条消息写入错误日志, 其中包含尝试装入的模块的名称以及来自操作系统的失败原因码。

这是一个输入字段。此字段的初始值为空字符串或空白。

CallbackType (MQLONG)

回调描述符结构- *CallbackType* 字段

这是回调函数的类型。值必须是下列其中一项:

MQCBT_MESSAGE_CONSUMER

将此回调定义为消息使用者函数。

当满足指定选择条件的消息在对象句柄上可用并启动连接时, 将调用消息使用者回调函数。

MQCBT_EVENT_HANDLER

将此回调定义为异步事件例程; 不驱动它使用句柄的消息。

在定义事件处理程序的 MQCB 调用上不需要 *Hobj*, 如果已指定, 那么将忽略。

将针对影响整个消息使用者环境的条件调用事件处理程序。当发生事件 (例如, 队列管理器或连接停止或停顿) 时, 将在没有消息的情况下调用使用者函数。对于特定于单个消息使用者的条件 (例如, MQRC_GET_禁止), 不会调用此参数。

事件将传递到应用程序, 无论连接是已启动还是已停止, 但在以下环境中除外:

- z/OS 环境上的 CICS
- 非线程应用程序

如果调用者未传递其中一个值, 那么调用将失败并返回 *Reason* 代码 MQRC_CALLBACK_TYPE_ERROR 这始终是一个输入字段。此字段的初始值为 MQCBT_MESSAGE_CONSUMER。

MaxMsg 长度 (MQLONG)

这是可以从句柄读取并提供给回调例程的最长消息的长度 (以字节计)。回调描述符结构- MaxMsg 长度字段如果消息的长度较长, 那么回调例程将接收 *MaxMsgLength* 字节的消息以及原因码:

- MQRC_TRUNCATED_MSG_FAILED 或
- 如果指定了 MQGMO_ACCEPT_TRUNCATED_MSG, 那么 MQRC_TRUNCATED_MSG_ACCEPT。

实际消息长度在 MQCBC 结构的 [第 253 页的『DataLength \(MQLONG\)』](#) 字段中提供。

定义了以下特殊值:

MQCBD_FULL_MSG_LENGTH

系统调整缓冲区长度以返回消息而不截断。

如果没有足够的内存可用于分配缓冲区以接收消息, 那么系统将使用 MQRC_STORAGE_NOT_AVAILABLE 原因码来调用回调函数。

例如, 如果您请求数据转换, 但没有足够的内存可用于转换消息数据, 那么未转换的消息将传递到回调函数。

这是一个输入字段。 *MaxMsgLength* 字段的初始值为 MQCBD_FULL_MSG_LENGTH。

选项 (MQLONG)

回调描述符结构-"选项" 字段

可以指定以下任何一项或全部内容。如果需要多个选项, 那么值可以是:

- 一起添加 (请勿多次添加同一常量), 或者
- 通过按位“或”运算组合在一起 (如果编程语言支持位运算)。

MQCBDO_FAIL_IF QUIESCING

如果队列管理器处于停顿状态, 那么 MQCB 调用将失败。

在 z/OS 上, 如果连接 (对于 CICS 或 IMS 应用程序) 处于停顿状态, 那么此选项还会强制 MQCB 调用失败。

在 MQCB 调用上传递的 MQGMO 选项中指定 MQGMO_FAIL_IF QUIESCING, 以在消息使用者停顿向其发出通知。

控制选项: 以下选项控制在使用者的状态发生更改时是否调用回调函数 (不带消息):

MQCBDO_REGISTER_CALL

回调函数通过调用类型 MQCBCT_REGISTER_CALL 进行调用。

MQCBDO_START_CALL

回调函数是使用调用类型 MQCBCT_START_CALL 调用的。

MQCBDO_STOP_CALL

回调函数通过调用类型 MQCBCT_STOP_CALL 进行调用。

MQCBDO_DEREGISTER_CALL

回调函数通过调用类型 MQCBCT_DEREGISTER_CALL 进行调用。

MQCBDO_EVENT_CALL

使用调用类型 MQCBCT_EVENT_CALL 调用回调函数。

MQCBDO_MC_EVENT_CALL

回调函数通过调用类型 MQCBCT_MC_EVENT_CALL 进行调用。

请参阅 第 252 页的『CallType (MQLONG)』，以获取有关这些调用类型的更多详细信息。

缺省选项: 如果不需要任何描述的选项，请使用以下选项：

MQCBDO_NONE

使用此值来指示未指定任何其他选项；所有选项均采用其缺省值。

MQCBDO_NONE 定义为帮助程序文档；不打算将此选项与任何其他选项一起使用，但由于其值为零，因此无法检测到此类使用。

这是一个输入字段。Options 字段的初始值为 MQCBDO_NONE。

StrucId (MQCHAR4)

回调描述符结构- StrucId 字段

这是结构标识；值必须为：

MQCBD_STRUC_ID

回调描述符结构的标识。

对于 C 编程语言，还定义了常量 MQCBD_STRUC_ID_ARRAY；此值与 MQCBD_STRUC_ID 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQCBD_STRUC_ID。

Version (MQLONG)

回调描述符结构-版本字段

这是结构版本号；值必须为：

MQCBD_VERSION_1

Version-1 回调描述符结构。

以下常量指定当前版本的版本号：

MQCBD_CURRENT_VERSION

当前版本的回调描述符结构。

这始终是一个输入字段。此字段的初始值为 MQCBD_VERSION_1。

MQCBD 的初始值和语言声明

回调描述符结构-初始值

| 表 481: MQCBD 中字段的初始值 | | |
|----------------------|------------------------|---------|
| 字段名称 | 常量的名称 | 常量值 |
| StrucId | MQCBD_STRUC_ID | 'CBD-' |
| Version | MQCBD_VERSION_1 | 1 |
| CallBackType | MQCBT_MESSAGE_CONSUMER | 1 |
| Options | MQCBDO_NONE | 0 |
| CallBackArea | None | 空指针或空空白 |
| CallBackFunction | None | 空指针或空空白 |
| CallBackName | None | 空字符串或空白 |
| MaxMsgLength | MQCBD_FULL_MSG_LENGTH | -1 |

表 481: MQCBD 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|--|-------|-----|
| 注意: | | |
| 1. 符号 \ 表示单个空白字符。 | | |
| 2. 值 "空字符串" 或 "空白" 表示 C 编程语言中的空字符串和其他编程语言中的空白字符。 | | |
| 3. 在 C 编程语言中, 宏变量 MQCBD_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值: | | |
| <pre>MQCBD MyCBD = {MQCBD_DEFAULT};</pre> | | |

C 声明

回调描述符结构-C 语言声明

```
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 声明

```
** MQCBD 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.
```

PL/I 声明

```
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-变量长度字符串

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|------------------|--|---------------------------|
| <i>VSPtr</i> | 指向可变长度字符串的指针 | VSPtr |
| <i>VSOffset</i> | 从包含此 MQCHARV 结构的结构开始的可变长度字符串的偏移量 (以字节为单位) | VSOffset |
| <i>VSLength</i> | 由 <i>VSPtr</i> 或 <i>VSOffset</i> 字段寻址的可变长度字符串的长度 (以字节为单位)。 | VSL 思 |
| <i>VSBufSize</i> | 由 <i>VSPtr</i> 或 <i>VSOffset</i> 字段寻址的缓冲区大小 (以字节计)。 | VSBufSize |
| <i>VSCCSID</i> | 由 <i>VSPtr</i> 或 <i>VSOffset</i> 字段寻址的可变长度字符串的字符集标识。 | VSCCSID |

MQCHARV 概述

可用性: AIX, HP-UX, Solaris, Linux, IBM i, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: 使用 MQCHARV 结构来描述可变长度字符串。

字符集和编码: MQCHARV 中的数据必须采用 MQENC_NATIVE 提供的本地队列管理器的编码以及结构中 VSCCSID 字段的字符集。如果应用程序作为 MQ 客户机运行, 那么结构必须采用客户机的编码。某些字符集具有依赖于编码的表示。如果 VSCCSID 是这些字符集之一, 那么所使用的编码与 MQCHARV 中其他字段的编码相同。由 VSCCSID 标识的字符集可以是双字节字符集 (DBCS)。

用法: MQCHARV 结构用于处理可能与包含它的结构不相邻的数据。要处理此数据, 可以使用使用指针数据类型声明的字段。请注意, COBOL 在所有环境中都不支持指针数据类型。因此, 也可以使用包含从包含 MQCHARV 的结构开始的数据偏移量的字段来对数据进行寻址。

COBOL 编程

如果要在环境之间移植应用程序, 那么必须确定指针数据类型是否在所有预期环境中都可用。否则, 应用程序必须使用偏移量字段而不是指针字段来寻址数据。

在不支持指针的环境中, 可以将指针字段声明为适当长度的字节字符串, 初始值为全空字节字符串。如果您正在使用偏移量字段, 请不要更改此初始值。在不更改提供的副本的情况下执行此操作的一种方法是使用以下内容:

```
COPY CMQCHRUV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

其中, 可以将 CMQCHRUV 交换为要使用的副本。

MQCHARV 的字段

MQCHARV 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

VSBufSize (MQLONG)

这是 *VSPtr* 或 *VSOffset* 字段寻址的缓冲区大小 (以字节计)。

当 MQCHARV 结构用作函数调用的输出字段时, 必须使用提供的缓冲区的长度来初始化此字段。如果 VSL 思的值大于 *VSBufSize*, 那么只会将 *VSBufSize* 字节的数据返回给缓冲区中的调用者。

此值必须是大于或等于零的值, 或者可识别的以下特殊值:

MQVS_USE_VSLENGTH

指定时, 将从 MQCHARV 结构中的 VSL 思长度字段获取缓冲区的长度。当使用结构作为输出字段并且提供了缓冲区时, 请勿使用此值。

这是此字段的初始值。

VSCCSID (MQLONG)

这是由 *VSPtr* 或 *VSOOffset* 字段寻址的可变长度字符串的字符集标识。

此字段的初始值为 *MQCCSI_APPL*，由 *MQ* 定义，以指示应将其更改为当前进程的真实字符集标识。因此，值 *MQCCSI_APPL* 从不与可变长度字符串关联。可以通过为编译单元的常量 *MQCCSI_APPL* 定义不同的值来更改此字段的初始值，方法是应用程序的编程语言定义相应的方法。

VSL 思 (*MQLONG*)

由 *VSPtr* 或 *VSOOffset* 字段寻址的可变长度字符串的长度 (以字节为单位)。

此字段的初始值为 0。该值必须大于或等于零或以下可识别的特殊值:

MQVS_NULL_TERMINATED

如果未指定 *MQVS_NULL_TERMINATED*，那么 *VSL* 思长度字节将包含在字符串中。如果存在空字符，那么不会对字符串进行定界。

如果指定了 *MQVS_NULL_TERMINATED*，那么字符串由字符串中迁到的第一个空值定界。空本身不包括在该字符串中。

注: 如果指定了 *MQVS_NULL_TERMINATED*，那么用于终止字符串的空字符是 *VSCCSID* 指定的代码集的空字符。

例如，在 UTF-16 (UCS-2 CCSID 1200 和 13488) 中，这是两个字节的 Unicode 编码，其中 null 由 16 位全零表示。在 UTF-16 中，通常会找到设置为全部为零的单个字节，这些字节是字符的一部分 (例如，7 位 ASCII 字符)，但仅当在偶数字节边界上找到两个 "零" 字节时，字符串才会以 null 结束。当两个 "零" 字节是有效字符的每个部分时，可以在奇数边界上获取两个 "零" 字节。例如，`x'01'x'00x'00'x'30'` 表示两个有效 Unicode 字符，并且不为空终止字符串。

VSOOffset (MQLONG)

偏移可以是正数或负数。您可以使用 *VSPtr* 或 *VSOOffset* 字段来指定可变长度字符串，但不能同时指定两者。从 *MQCHARV* 的开头或包含它的结构开始的可变长度字符串的偏移量 (以字节为单位)。

当 *MQCHARV* 结构嵌入到另一个结构中时，此值是包含此 *MQCHARV* 结构的结构开头的可变长度字符串的偏移量 (以字节为单位)。当 *MQCHARV* 结构未嵌入在另一个结构中时，例如，如果将其指定为函数调用上的参数，那么偏移量相对于 *MQCHARV* 结构的开始。

此字段的初始值为 0。

VSPtr (MQPTR)

这是指向可变长度字符串的指针。

您可以使用 *VSPtr* 或 *VSOOffset* 字段来指定可变长度字符串，但不能同时指定两者。

此字段的初始值为空指针或空字节。

***MQCHARV* 的初始值和语言声明**

***MQCHARV* 中字段的初始值**

| 字段名称 | 常量的名称 | 常量值 |
|------------------|--------------------------|----------|
| <i>VSPtr</i> | None | 空指针或空字节。 |
| <i>VSOOffset</i> | None | 0 |
| <i>VSBufSize</i> | <i>MQVS_USE_VSLENGTH</i> | 0 |
| <i>VSLength</i> | None | 0 |
| <i>VSCCSID</i> | <i>MQCCSI_APPL</i> | -3 |

| 字段名称 | 常量的名称 | 常量值 |
|---|-------|-----|
| <p>注: 在 C 编程语言中, 宏变量 MQCHARV_DEFAULT 包含上面列出的值。它可以通过以下方式用于为结构中的字段提供初始值:</p> <pre>MQCHARV MyVarStr = {MQCHARV_DEFAULT};</pre> | | |

C 声明

```
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 */
};
```

MQCHARV 的 COBOL 声明

```
** 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.
```

PL/I 声明

```
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 */
```

High Level Assembler 声明

```
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
MQCHARV_AREA     DS    CL(MQCHARV_LENGTH)
```

重新定义 MQCCSI_APPL

以下示例显示如何以各种编程语言覆盖 MQCCSI_APPL 的值。您可以更改 MQCCSI_APPL 的值, 从而无需单独为每个变长字符串设置 VSCCSID。

在这些示例中, CCSID 设置为 1208; 将此值更改为所需的值。这将成为缺省值, 您可以通过在 MQCHARV 的任何特定实例中设置 VSCCSID 来覆盖该值。

C 用途

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

COBOL 用途

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

PL/I 使用情况

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

System/390 汇编程序用法

```
MQCCSI_APPL EQU 1208
CMQA LIST=NO
```

MQCIH- CICS 网桥头

下表汇总了结构中的字段。

| 表 482: MQCIH 中的字段 | | |
|---------------------------|-----------------------------|------------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>StrucLength</i> | MQCIH 结构的长度 | StrucLength |
| <i>Encoding</i> | 已保留 | 编码 |
| <i>CodedCharSetId</i> | 已保留 | CodedCharSetId |
| <i>Format</i> | MQCIH 之后数据的 MQ 格式名称 | FORMAT |
| <i>Flags</i> | 标志 | 标志 |
| <i>ReturnCode</i> | 来自网桥的返回码 | ReturnCode |
| <i>CompCode</i> | MQ 完成代码或 CICS EIBRESP | CompCode |
| <i>Reason</i> | MQ 原因或反馈代码, 或 CICS EIBRESP2 | 原因 |
| <i>UOWControl</i> | 工作单元控件 | UOWControl |
| <i>GetWaitInterval</i> | 由网桥任务发出的 MQGET 调用的等待时间间隔 | GetWaitInterval |
| <i>LinkType</i> | 链接类型 | LinkType |
| <i>OutputDataLength</i> | 输出 COMMAREA 数据长度 | OutputDataLength |
| <i>FacilityKeepTime</i> | 网桥设施发行时间 | FacilityKeepTime |
| <i>ADSDescriptor</i> | 发送/接收 ADS 描述符 | ADSDescriptor |
| <i>ConversationalTask</i> | 任务是否可为会话式任务 | ConversationalTask |
| <i>TaskEndStatus</i> | 任务结束时的状态 | TaskEndStatus |
| <i>Facility</i> | 网桥设施令牌 | 设施 |

表 482: MQCIH 中的字段 (继续)

| 字段 | 描述 | Topic |
|---|------------------------|-----------------------------------|
| <i>Function</i> | MQ 调用名称或 CICS EIBFN 函数 | 函数 |
| <i>AbendCode</i> | 异常终止代码 | AbendCode |
| <i>Authenticator</i> | 密码或通行票 | Authenticator |
| <i>Reserved1</i> | 已保留 | Reserved1 |
| <i>ReplyToFormat</i> | 应答消息的 MQ 格式名称 | ReplyToFormat |
| <i>RemoteSysId</i> | 要使用的远程 CICS 系统标识 | RemoteSysId |
| <i>RemoteTransId</i> | 要使用的 CICS RTRANSID | RemoteTransId |
| <i>TransactionId</i> | 要附加的事务 | TransactionId |
| <i>FacilityLike</i> | 终端仿真属性 | FacilityLike |
| <i>AttentionId</i> | AID 密钥 | AttentionId |
| <i>StartCode</i> | 事务开始代码 | StartCode |
| <i>CancelCode</i> | 异常结束事务代码 | CancelCode |
| <i>NextTransactionId</i> | 要附加的下一个事务 | NextTransactionId |
| <i>Reserved2</i> | 已保留 | Reserved2 |
| <i>Reserved3</i> | 已保留 | Reserved3 |
| 注: 如果 <i>Version</i> 小于 MQCIH_VERSION_2, 那么不存在其余字段。 | | |
| <i>CursorPosition</i> | 光标位置 | CursorPosition |
| <i>ErrorOffset</i> | 消息中错误的偏移量 | ErrorOffset |
| <i>InputItem</i> | 已保留 | InputItem |
| <i>Reserved4</i> | 已保留 | Reserved4 |

MQCIH 概述

可用性: AIX, HP-UX, z/OS, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: MQCIH 结构描述在通过 WebSphere MQ for z/OS 发送到 CICS 网桥的消息开始时可能存在的信息。

格式名称: MQFMT_CICS。

版本: MQCIH 的当前版本为 MQCIH_VERSION_2。仅在结构的最新版本中存在的字段将在随后的描述中标识为此类字段。

为受支持编程语言提供的头 COPY 和 INCLUDE 文件包含最新版本的 MQCIH, *Version* 字段的初始值设置为 MQCIH_VERSION_2。

字符集和编码: 特殊条件适用于用于 MQCIH 结构和应用程序消息数据的字符集和编码:

- 连接到拥有 CICS 网桥队列的队列管理器的应用程序必须提供队列管理器的字符集和编码中的 MQCIH 结构。这是因为在这种情况下不会执行 MQCIH 结构的数据转换。
- 连接到其他队列管理器的应用程序可以提供 MQCIH 结构, 该结构位于任何受支持的字符集和编码中; 连接到拥有 CICS 网桥队列的队列管理器的接收消息通道代理将转换 MQCIH 结构。
- 遵循 MQCIH 结构的应用程序消息数据必须采用与 MQCIH 结构相同的字符集和编码。不能使用 MQCIH 结构中的 *CodedCharSetId* 和 *Encoding* 字段来指定应用程序消息数据的字符集和编码。

如果数据不是队列管理器支持的其中一种内置格式，那么必须提供数据转换出口以转换应用程序消息数据。

用法: 如果应用程序需要与第 276 页的表 484 中显示的初始值相同的值，并且网桥正在使用 AUTH=LOCAL 或 AUTH=IDENTIFY 运行，那么可以从消息中省略 MQCIH 结构。在所有其他情况下，该结构必须存在。

网桥接受 version-1 或 version-2 MQCIH 结构，但对于 3270 事务，必须使用 version-2 结构。

应用程序必须确保记录为请求字段的字段在发送到网桥的消息中具有相应的值；这些字段是对网桥的输入。

记录为响应字段的字段由 CICS 网桥在网桥发送到应用程序的应答消息中设置。将在 *ReturnCode*，*Function*，*CompCode*，*Reason* 和 *AbendCode* 字段中返回错误信息，但并非所有这些错误信息都是在所有情况下设置的。第 268 页的表 483 显示了为 *ReturnCode* 的不同值设置的字段。

| ReturnCode | Function | CompCode | Reason | AbendCode |
|---|-----------------|-----------------|---------------|------------------|
| MQCRC_OK | - | - | - | - |
| MQCRC_BRIDGE_ERROR | - | - | MQFB_CICS_* | - |
| MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT | MQ 调用名称 | 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 |

MQCIH 的字段

MQCIH 结构包含以下字段；这些字段以字母顺序描述：

AbendCode (MQCHAR4)

AbendCode 是响应字段。此字段的长度由 MQ_ABEND_CODE_LENGTH 指定。此字段的初始值为 4 个空白字符。

仅当 *ReturnCode* 字段具有值 MQCRC_APPLICATION_ABEND 或 MQCRC_BRIDGE_ABEND 时，此字段中返回的值才有意义。如果存在，那么 *AbendCode* 将包含 CICS ABCODE 值。

ADSDescriptor (MQLONG)

此字段是一个指示符，用于指定是否在 SEND 和 RECEIVE BMS 请求上发送 ADS 描述符。

已定义下列值：

MQCADSD_NONE

不发送或接收 ADS 描述符。

MQCADSD_SEND

发送 ADS 描述符。

MQCADSD_RECV

接收 ADS 描述符。

MQCADSD_MSGFORMAT

对 ADS 描述符使用消息格式。

这将使用 ADS 描述符的长格式发送或接收 ADS 描述符。长格式具有在 4 字节边界上对齐的字段。

按如下所示设置 *ADSDescriptor* 字段：

- 如果不使用 ADS 描述符，请将该字段设置为 MQCADSD_NONE。

- 如果在每个环境中使用具有相同 CCSID 的 ADS 描述符, 请将该字段设置为 MQCADSD_SEND 和 MQCADSD_RECV 的总和。
- 如果要在每个环境中使用具有不同 CCSID 的 ADS 描述符, 请将该字段设置为 MQCADSD_SEND, MQCADSD_RECV 和 MQCADSD_MSGFORMAT 的总和。

这是仅用于 3270 事务的请求字段。此字段的初始值为 MQCADSD_NONE。

AttentionId (MQCHAR4)

此字段中的值确定启动事务时 AID 键的初始值。它是左对齐的 1 字节值。

AttentionId 是仅用于 3270 事务的请求字段。此字段的长度由 MQ_ATTENTION_ID_LENGTH 指定。此字段的初始值为 4 个空格。

鉴别符 (MQCHAR8)

此字段的值是密码或通行票。

如果针对 CICS 网桥的用户标识认证处于活动状态, 那么 *Authenticator* 将与 MQMD 身份上下文中的用户标识配合使用, 以认证消息的发送方。

这是请求字段。此字段的长度由 MQ_AUTHENTICATOR_LENGTH 指定。此字段的初始值为 8 空白。

CancelCode (MQCHAR4)

此字段中的值是要用于终止事务 (通常是请求更多数据的会话式事务) 的异常终止代码。否则, 此字段设置为空白。

此字段是仅用于 3270 事务的请求字段。此字段的长度由 MQ_CANCEL_CODE_LENGTH 给出。此字段的初始值为 4 个空格。

CodedCharSetId (MQLONG)

CodedCharSetId 是保留字段; 其值不重要。此字段的初始值为 0。

遵循 MQCIH 结构的受支持结构的字符集标识与 MQCIH 结构本身的字符集标识相同, 并且取自任何先前的 WebSphere MQ 头。

CompCode (MQLONG)

此字段是响应字段。其初始值为 MQCC_OK

此字段中返回的值取决于 *ReturnCode*; 请参阅 [第 268 页的表 483](#)。

ConversationalTask (MQLONG)

此字段是一个指示符, 用于指定是允许任务发出请求以获取更多信息, 还是停止任务并发出异常终止消息。

该值必须是下列其中一个选项:

MQCCT_YES

该任务是对话式任务。

MQCCT_NO

该任务不是会话式任务。

此字段是仅用于 3270 事务的请求字段。此字段的初始值为 MQCCT_NO。

CursorPosition (MQLONG)

此字段中的值显示事务启动时的初始光标位置。对于会话式事务, 光标位置在 RECEIVE 向量中。

此字段是仅用于 3270 事务的请求字段。此字段的初始值为 0。如果 *Version* 小于 MQCIH_VERSION_2, 那么此字段不存在。

Encoding (MQLONG)

此字段是保留字段; 其值不重要。其初始值为 0。

遵循 MQCIH 结构的受支持结构的编码与 MQCIH 结构本身的编码相同, 取自任何先前的 WebSphere MQ 头。

ErrorOffset (MQLONG)

ErrorOffset 字段显示网桥出口检测到的无效数据的位置。此字段提供从消息开始到无效数据位置的偏移量。

ErrorOffset 是仅用于 3270 事务的响应字段。此字段的初始值为 0。如果 *Version* 小于 MQCIH_VERSION_2, 那么此字段不存在。

设施 (MQBYTE8)

此字段显示 8 字节网桥设施令牌。

网桥设施令牌使伪对话中的多个事务能够使用同一网桥设施 (虚拟 3270 终端)。在伪对话中的第一条或仅一条消息中, 设置 MQCFAC_NONE 值。此值指示 CICS 为此消息分配新的网桥设施。当在输入消息上指定非零 *FacilityKeepTime* 时, 将在响应消息中返回网桥设施令牌。然后, 伪对话中的后续输入消息必须使用相同的网桥设施令牌。

定义了以下特殊值:

MQCFAC_NONE

未指定设施令牌。

对于 C 编程语言, 还定义了常量 MQCFAC_NONE_ARRAY, 其值与 MQCFAC_NONE 相同, 但它是字符数组而不是字符串。

此字段既是请求字段, 也是仅用于 3270 事务的响应字段。此字段的长度由 MQ_FACILITY_LENGTH 提供。此字段的初始值为 MQCFAC_NONE。

FacilityKeep 时间 (MQLONG)

FacilityKeep 时间是用户事务结束后网桥设施保留的时间长度 (以秒为单位)。

对于伪会话式事务, 请指定与伪会话的预期持续时间相对应的值; 对于伪会话的最后一个事务, 请指定零, 对于其他事务类型, 请指定零。

此字段是仅用于 3270 事务的请求字段。此字段的初始值为 0。

FacilityLike (MQCHAR4)

FacilityLike 是要用作网桥设施模型的已安装终端的名称。

空白值表示从网桥事务概要文件定义中获取 *FacilityLike*, 或者使用缺省值。

此字段是仅用于 3270 事务的请求字段。此字段的长度由 MQ_FACILITY_LIKE_LENGTH 提供。此字段的初始值为 4 个空格。

Flags (MQLONG)

此字段是请求字段。此字段的初始值为 MQCIH_NONE。

该值必须为:

MQCIH_NONE

没有标志。

MQCIH_PASS_EXPIRATION

应答消息包含:

- 与请求消息相同的到期报告选项。
- 请求消息中没有对网桥处理时间进行调整的剩余到期时间。

如果省略此值, 那么到期时间将设置为 无限制。

MQCIH_REPLY_WITHOUT_NULLS

调整 CICS DPL 程序请求的应答消息长度, 以排除 DPL 程序返回的 COMMAREA 末尾的尾部空值 (X'00')。如果未设置此值, 那么空值可能很重要, 并且将返回完整的 COMMAREA。

MQCIH_SYNC_ON_RETURN

用于 DPL 请求的 CICS 链接使用 SYNCONRETURN 选项, 这将导致 CICS 在程序完成时获取同步点 (如果该程序已交付到另一个 CICS 区域)。网桥未指定要向哪个 CICS 区域发送请求; 该请求由 CICS 程序定义或工作负载均衡设施控制。

Format (MQCHAR8)

此字段显示遵循 MQCIH 结构的数据的 WebSphere MQ 格式名称。

在 MQPUT 或 MQPUT1 调用上，应用程序必须将此字段设置为适合于数据的值。此字段的编码规则与 MQMD 中 *Format* 字段的编码规则相同。

如果 *ReplyToFormat* 字段的值为 MQFMT_NONE，那么此格式名称也将用于应答消息。

- 对于 DPL 请求，*Format* 必须是 COMMAREA 的格式名。
- 对于 3270 请求，*Format* 必须是 CSQCBDCI，并且网桥将应答消息的格式设置为 CSQCBDCO。

这些格式的数据转换出口必须安装在要运行这些格式的队列管理器上。

如果请求消息生成错误应答消息，那么该错误应答消息的格式名称为 MQFMT_STRING。

此字段是请求字段。此字段的长度由 MQ_FORMAT_LENGTH 指定。此字段的初始值为 MQFMT_NONE。

函数 (MQCHAR4)

此字段是响应字段。此字段的长度由 MQ_FUNCTION_LENGTH 给出。此字段的初始值为 MQCFUNC_NONE。

此字段中返回的值取决于 *ReturnCode*；请参阅第 268 页的表 483。当 *Function* 包含 WebSphere MQ 调用名称时，可以使用以下值：

MQCFUNC_MQCONN

MQCONN 调用。

MQCFUNC_MQGET

MQGET 调用。

MQCFUNC_MQINQ

MQINQ 调用。

MQCFUNC_MQOPEN

MQOPEN 调用。

MQCFUNC_MQPUT

MQPUT 调用。

MQCFUNC_MQPUT1

MQPUT1 调用。

MQCFUNC_NONE

没有电话

在所有情况下，对于 C 编程语言，还定义了常量 MQCFUNC_*_ARRAY；这些常量具有与相应的 MQCFUNC_* 常量相同的值，但是字符数组而不是字符串。

GetWait 时间间隔 (MQLONG)

此字段是请求字段。其初始值为 MQCGWI_DEFAULT。

仅当 *UOWControl* 具有值 MQCUOWC_FIRST 时，此字段才适用。它使发送应用程序能够指定网桥发出的 MQGET 调用将等待由此消息启动的工作单元的第二条和后续请求消息的近似时间（以毫秒计）。此设施将覆盖网桥使用的缺省等待时间间隔。您可以使用以下特殊值：

MQCGWI_DEFAULT

缺省等待时间间隔。

此值会导致 CICS 网桥等待该网桥启动时指定的时间。

MQWI_UNLIMITED

无限制的等待时间间隔。

InputItem (MQLONG)

此字段是保留字段。该值必须为 0。

如果 *Version* 小于 MQCIH_VERSION_2，那么此字段不存在。

LinkType (MQLONG)

此字段是请求字段。其初始值为 MQCLT_PROGRAM。

此值指示网桥尝试链接的对象类型。它必须是下列其中一个值：

MQCLT_PROGRAM

DPL 程序。

MQCLT_TRANSACTION

3270 事务。

NextTransaction 标识 (MQCHAR4)

此值是用户事务 (通常由 EXEC CICS RETURN TRANSID) 返回的下一个事务的名称。如果没有下一个事务, 那么此字段设置为空白。

此字段是仅用于 3270 事务的响应字段。此字段的长度由 MQ_TRANSACTION_ID_LENGTH 指定。此字段的初始值为 4 个空格。

OutputData 长度 (MQLONG)

此字段是仅用于 DPL 程序的请求字段。其初始值为 MQCODL_AS_INPUT。

此值是要在应答消息中返回到客户机的用户数据的长度。此长度包含 8 字节的程序名。传递给链接程序的 COMMAREA 的长度是此字段的最大长度以及请求消息中用户数据的长度减去 8。

注: 消息中用户数据的长度是不包含 MQCIH 结构的消息的长度。

如果请求消息中的用户数据长度小于 *OutputDataLength*, 那么将使用 LINK 命令的 DATALENGTH 选项, 从而使 LINK 能够高效地将功能交付到另一个 CICS 区域。

可以使用以下特殊值:

MQCODL_AS_INPUT

输出长度与输入长度相同。

即使不请求应答, 也可能需要此值, 以确保传递给链接程序的 COMMAREA 具有足够的大小。

原因 (MQLONG)

此字段是响应字段。其初始值为 MQRC_NONE。

此字段中返回的值取决于 *ReturnCode*; 请参阅 [第 268 页的表 483](#)。

RemoteSys 标识 (MQCHAR4)

此字段显示处理请求的 CICS 系统的 CICS 系统标识。

如果此字段为空, 那么将在与网桥监视器相同的 CICS 系统上处理 CICS 系统请求。在 "应答" 消息中返回了所使用的 SYSID。

对于 3270 伪对话, 对话中的所有后续消息都必须指定初始应答中返回的远程 SYSID。如果指定了此参数, 那么 SYSID 必须:

- 保持活动状态。
- 有权访问 WebSphere MQ 请求队列。
- 可由网桥监视器的 CICS 系统中的 CICS ISC 链接访问。

RemoteTrans 标识 (MQCHAR4)

此字段是可选的 "请求" 字段。此字段的长度由 MQ_TRANSACTION_ID_LENGTH 指定。

如果指定了此字段, 那么此字段将用作 CICS START 的 RTRANSID 值。

ReplyTo 格式 (MQCHAR8)

此字段的值是响应当前消息而发送的应答消息的 WebSphere MQ 格式名称。

用于对此字段进行编码的规则与用于在 MQMD 中对 *Format* 字段进行编码的规则相同。

此字段是仅用于 DPL 程序的请求字段。此字段的长度由 MQ_FORMAT_LENGTH 指定。此字段的初始值为 MQFMT_NONE。

Reserved1 (MQCHAR8)

此字段是保留字段。该值必须为 8 空白。

Reserved2 (MQCHAR8)

此字段是保留字段。该值必须为 8 空白。

Reserved3 (MQCHAR8)

此字段是保留字段。该值必须为 8 空白。

Reserved4 (MQLONG)

此字段是保留字段。该值必须为 0。

如果 *Version* 小于 MQCIH_VERSION_2, 那么此字段不存在。

ReturnCode (MQLONG)

此字段的值是来自 CICS 网桥的返回码, 用于描述该网桥执行的处理的结果。此字段是响应字段, 初始值为 MQCRC_OK。

Function, *CompCode*, *Reason* 和 *AbendCode* 字段可能包含其他信息 (请参阅 [第 268 页的表 483](#))。该值为下列其中之一:

MQCRC_APPLICATION_ABEND

(5, X'005 ') 应用程序异常结束。

MQCRC_BRIDGE_ABEND

(4, X'004 ') CICS 网桥异常结束。

MQCRC_BRIDGE_ERROR

(3, X'003 ') CICS 网桥检测到错误。

MQCRC_BRIDGE_TIMEOUT

(8, X'008 ') 在指定时间内未接收到当前工作单元中的第二条或更高版本的消息。

MQCRC_CICS_EXEC_ERROR

(1, X'001 ') EXEC CICS 语句检测到错误。

MQCRC_MQ_API_ERROR

(2, X'002 ') MQ 调用检测到错误。

MQCRC_OK

(0, X'000 ') 无错误。

MQCRC_PROGRAM_NOT_AVAILABLE

(7, X'007 ') 程序不可用。

MQCRC_SECURITY_ERROR

(6, X'006 ') 发生安全性错误。

MQCRC_TRANSID_NOT_AVAILABLE

(9, X'009 ') 事务不可用。

StartCode (MQCHAR4)

此字段的值是一个指示符, 用于指定网桥是模拟终端事务还是使用 START 启动的事务。

值必须为以下其中一项:

MQCSC_START

开始。

MQCSC_STARTDATA

启动数据。

MQCSC_TERMINPUT

终端输入。

MQCSC_NONE

无。

在所有情况下，对于 C 编程语言，还定义了常量 MQCSC_*_ARRAY; 这些常量具有与相应 MQCSC_* 常量相同的值，但是字符数组而不是字符串。

在来自网桥的响应中，此字段设置为适合于 *NextTransactionId* 字段中包含的下一个事务标识的开始代码。响应中可能包含以下开始代码：

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT

对于 CICS Transaction Server V 1.2，此字段仅是请求字段；其在响应中的值未定义。

对于 CICS Transaction Server V 1.3 和后续发行版，此字段既是请求字段，也是响应字段。

此字段仅用于 3270 事务。此字段的长度由 MQ_START_CODE_LENGTH 给出。此字段的初始值为 MQCSC_NONE。

StrucId (MQCHAR4)

此字段是请求字段，初始值为 MQCIH_STRUC_ID。

该值必须为：

MQCIH_STRUC_ID

CICS 信息头结构的标识。

对于 C 编程语言，还定义了常量 MQCIH_STRUC_ID_ARRAY; 此值与 MQCIH_STRUC_ID 相同，但是字符数组而不是字符串。

StrucLength (MQLONG)

此字段是请求字段，初始值为 MQCIH_LENGTH_2。

值必须为以下其中一项：

MQCIH_LENGTH_1

version-1 CICS 信息头结构的长度。

MQCIH_LENGTH_2

version-2 CICS 信息头结构的长度。

以下常量指定当前版本的长度：

MQCIH_CURRENT_LENGTH

当前版本的 CICS 信息头结构的长度。

TaskEnd 状态 (MQLONG)

此字段是响应字段，显示任务结束时用户事务的状态。该字段仅用于 3270 事务，其初始值为 MQCTES_NOSYNC。

将返回下列其中一个值：

MQCTES_NOSYNC

未同步。

用户事务尚未完成，并且未同步。在此情况下，MQMD 中的 *MsgType* 字段为 MQMT_REQUEST。

MQCTES_COMMIT

落实工作单元。

用户事务尚未完成，但已同步第一个工作单元。在此情况下，MQMD 中的 *MsgType* 字段为 MQMT_DATAGRAM。

MQCTES_BACKOUT

回退工作单元。

用户事务尚未完成。当前工作单元已回退。在此情况下，MQMD 中的 *MsgType* 字段为 MQMT_DATAGRAM。

MQCTES_ENDTASK

结束任务。

用户事务已结束 (或异常结束)。在此情况下，MQMD 中的 *MsgType* 字段为 MQMT_REPLY。

TransactionId (MQCHAR4)

此字段是请求字段。其长度由 MQ_TRANSACTION_ID_LENGTH 指定。此字段的初始值为 4 个空格。

如果 *LinkType* 具有值 MQCLT_TRANSACTION，那么 *TransactionId* 是要运行的用户事务的事务标识；请在此情况下指定非空白值。

如果 *LinkType* 具有值 MQCLT_PROGRAM，那么 *TransactionId* 是用于运行工作单元中所有程序的事务代码。如果指定空白值，那么将使用 CICS DPL 网桥缺省事务代码 (CKBP)。如果该值为非空白，那么您必须将其定义为具有初始程序 CSQCBP00 的 CICS 本地事务。仅当 *UOWControl* 具有值 MQCUOWC_FIRST 或 MQCUOWC_ONLY 时，此字段才适用。

UOWControl (MQLONG)

此字段是一个请求字段，用于控制由 CICS 网桥执行的工作单元处理。此字段的初始值为 MQCUOWC_ONLY。

您可以请求网桥运行单个事务，或者在工作单元中运行一个或多个程序。该字段指示 CICS 网桥是启动工作单元，在当前工作单元中执行请求的功能，还是通过落实或回退工作单元来结束工作单元。支持各种组合，以优化数据传输流。

值必须为以下其中一项：

MQCUOWC_ONLY

启动工作单元，执行功能，然后落实工作单元。

MQCUOWC_CONTINUE

当前工作单元的附加数据 (仅限 3270)。

MQCUOWC_FIRST

启动工作单元并执行功能。

MQCUOWC_中间件

在当前工作单元中执行功能

MQCUOWC_LAST

执行函数，然后落实工作单元。

MQCUOWC_COMMIT

落实工作单元 (仅限 DPL)。

MQCUOWC_BACKOUT

回退工作单元 (仅限 DPL)。

Version (MQLONG)

此字段是请求字段。其初始值为 MQCIH_VERSION_2。

值必须为以下其中一项：

MQCIH_VERSION_1

Version-1 CICS 信息头结构。

MQCIH_VERSION_2

Version-2 CICS 信息头结构。

仅在结构的最新版本中存在的字段在字段描述中标识为此类字段。以下常量指定当前版本的版本号：

MQCIH_CURRENT_VERSION

CICS 信息头结构的当前版本。

MQCIH 的初始值和语言声明

| 表 484: MQCIH 的 MQCIH 中字段的初始值 | | |
|------------------------------|-----------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQCIH_STRUC_ID | 'CIH~' |
| <i>Version</i> | MQCIH_VERSION_2 | 2 |
| <i>StrucLength</i> | MQCIH_LENGTH_2 | 180 |
| <i>Encoding</i> | None | 0 |
| <i>CodedCharSetId</i> | None | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQCIH_NONE | 0 |
| <i>ReturnCode</i> | MQCRC_OK | 0 |
| <i>CompCode</i> | MQCC_OK | 0 |
| <i>Reason</i> | MQRC_NONE | 0 |
| <i>UOWControl</i> | MQCUOWC_ONLY | 273 |
| <i>GetWaitInterval</i> | MQCGWI_DEFAULT | -2 |
| <i>LinkType</i> | MQCLT_PROGRAM | 1 |
| <i>OutputDataLength</i> | MQCODL_AS_INPUT | -1 |
| <i>FacilityKeepTime</i> | None | 0 |
| <i>ADSDescriptor</i> | MQCADSD_NONE | 0 |
| <i>ConversationalTask</i> | MQCCT_NO | 0 |
| <i>TaskEndStatus</i> | MQCTES_NOSYNC | 0 |
| <i>Facility</i> | MQCFAC_NONE | Null |
| <i>Function</i> | MQCFUNC_NONE | 空白 |
| <i>AbendCode</i> | None | 空白 |
| <i>Authenticator</i> | None | 空白 |
| <i>Reserved1</i> | None | 空白 |
| <i>ReplyToFormat</i> | MQFMT_NONE | 空白 |
| <i>RemoteSysId</i> | None | 空白 |
| <i>RemoteTransId</i> | None | 空白 |
| <i>TransactionId</i> | None | 空白 |
| <i>FacilityLike</i> | None | 空白 |
| <i>AttentionId</i> | None | 空白 |
| <i>StartCode</i> | MQCSC_NONE | 空白 |
| <i>CancelCode</i> | None | 空白 |
| <i>NextTransactionId</i> | None | 空白 |
| <i>Reserved2</i> | None | 空白 |

表 484: MQCIH 的 MQCIH 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|-----------------------|-------|-----|
| <i>Reserved3</i> | None | 空白 |
| <i>CursorPosition</i> | None | 0 |
| <i>ErrorOffset</i> | None | 0 |
| <i>InputItem</i> | None | 0 |
| <i>Reserved4</i> | None | 0 |

注意:

1. 符号 `~` 表示单个空白字符。
2. 在 C 编程语言中, 宏变量 `MQCIH_DEFAULT` 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQCIH MyCIH = {MQCIH_DEFAULT};
```

C 声明

```
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 声明

```

** 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.

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

| | | | |
|--------------------------|-------|-----|--|
| MQCIH | DSECT | | |
| MQCIH_STRUCID | DS | CL4 | Structure identifier |
| MQCIH_VERSION | DS | F | Structure version number |
| MQCIH_STRUCLNGTH | 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_ADSDSCRIPTOR | 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) | |

Visual Basic 声明

```

Type MQCIH
  StructId      As String*4 'Structure identifier'
  Version       As Long     'Structure version number'
  StructLength  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-创建消息句柄选项

下表汇总了结构中的字段。

表 485: MQCMHO 中的字段

| 字段 | 描述 | Topic |
|----------------|-------|-------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 选项 | 选项 |

MQCMHO 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS 和 WebSphere MQ 客户机。

用途: MQCMHO 结构允许应用程序指定用于控制如何创建消息句柄的选项。该结构是 MQCRTMH 调用上的输入参数。

字符集和编码: MQCMHO 中的数据必须位于应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQCMHO 的字段

MQCMHO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

选项 (MQLONG)

此字段始终是输入字段。其初始值为 MQCMHO_DEFAULT_VALIDATION。

可以指定下列其中一个选项:

MQCMHO_VALIDATE

当调用 MQSETMP 以在此消息句柄中设置属性时, 将验证属性名称以确保它:

- 不包含无效字符。
- 不开始 JMS 或 usr.JMS, 但下列情况除外:

- JMSCorrelationID
- JMSReplyTo
- JMSType
- JMSXGroupID
- JMSXGroupSeq

这些名称保留用于 JMS 属性。

- 不是下列其中一个关键字 (大小写任意混合):

- AND
- BETWEEN
- ESCAPE
- FALSE
- IN
- IS
- LIKE
- 非
- NULL
- 或者
- TRUE

- 未开始正文。或根。(Root.MQMD 除外。)

如果该属性是 MQ 定义的 (mq. *) 并且识别名称, 将属性描述符字段设置为属性的正确值。如果无法识别该属性, 那么属性描述符的 *Support* 字段将设置为 **MQPD_OPTIONAL**。

MQCMHO_DEFAULT_VALIDATION

此值指定对属性名进行缺省级别的验证。

缺省验证级别等同于 **MQCMHO_VALIDATE** 指定的级别。

该值为缺省值。

MQCMHO_NO_VALIDATION

不会对属性名进行验证。请参阅 **MQCMHO_VALIDATE** 的描述。

缺省选项: 如果不需要上述任何选项, 那么可以使用以下选项:

MQCMHO_NONE

所有选项都采用其缺省值。使用此值可指示未指定任何其他选项。**MQCMHO_NONE** 帮助程序文档; 不打算将此选项与任何其他选项一起使用, 但由于其值为零, 因此无法检测到此类使用。

StrucId (MQCHAR4)

此字段始终是输入字段。其初始值为 **MQCMHO_STRUC_ID**。

这是结构标识; 值必须为:

MQCMHO_STRUC_ID

创建消息句柄选项结构的标识。

对于 C 编程语言, 还定义了常量 **MQCMHO_STRUC_ID_ARRAY**; 此值与 **MQCMHO_STRUC_ID** 相同, 但是字符数组而不是字符串。

Version (MQLONG)

此字段始终是输入字段。其初始值为 **MQCMHO_VERSION_1**。

这是结构版本号; 值必须为:

MQCMHO_VERSION_1

Version-1 创建消息句柄选项结构。

以下常量指定当前版本的版本号:

MQCMHO_CURRENT_VERSION

当前版本的创建消息句柄选项结构。

MQCMHO 的初始值和语言声明

| 表 486: MQCMHO 中字段的初始值 | | |
|---|----------------------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQCMHO_STRUC_ID | 'CMHO' |
| <i>Version</i> | MQCMHO_VERSION_1 | 1 |
| <i>Options</i> | MQCMHO_DEFAULT_VALIDATION | 0 |
| <p>注意:</p> <p>1. 在 C 编程语言中, 宏变量 MQCMHO_DEFAULT 包含上面列出的值。它可以通过以下方式用于为结构中的字段提供初始值:</p> <pre>MQCMHO MyCMHO = {MQCMHO_DEFAULT};</pre> | | |

C 声明

```
struct tagMQCMHO {
    MQCHAR4  StrucId;          /* Structure identifier */
    MQLONG   Version;         /* Structure version number */
    MQLONG   Options;        /* Options that control the action of MQCRTMH */
};
```

COBOL 声明

```
** 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.
```

PL/I 声明

```
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 */
```

High Level Assembler 声明

```
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 - 连接选项

下表汇总了结构中的字段。

| 表 487: MQCNO 中的字段 | | |
|---|--------------------|--------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 用于控制 MQCONN 的操作的选项 | 选项 |
| 注: 如果 <i>Version</i> 小于 MQCNO_VERSION_2, 那么将忽略其余字段。 | | |
| <i>ClientConnOffset</i> | 客户机连接的 MQCD 结构的偏移量 | ClientConn 偏移量 |
| <i>ClientConnPtr</i> | 客户机连接的 MQCD 结构的地址 | ClientConnPtr |
| 注: 如果 <i>Version</i> 小于 MQCNO_VERSION_3, 那么将忽略其余字段。 | | |
| <i>ConnTag</i> | 队列管理器连接标记 | ConnTag |
| 注: 如果 <i>Version</i> 小于 MQCNO_VERSION_4, 那么将忽略其余字段。 | | |

表 487: MQCNO 中的字段 (继续)

| 字段 | 描述 | Topic |
|---|---------------------|-----------------------------------|
| <i>SSLConfigPtr</i> | 客户机连接的 MQSCO 结构的地址 | SSLConfigPtr |
| <i>SSLConfigOffset</i> | 客户机连接的 MQSCO 结构的偏移量 | SSLConfigOffset |
| 注: 如果 <i>Version</i> 小于 MQCNO_VERSION_5, 那么将忽略其余字段。 | | |
| <i>ConnectionId</i> | 唯一连接标识 | ConnectionId |
| <i>SecurityParmsOffset</i> | 安全性参数 | SecurityParms 偏移量 |
| <i>SecurityParmsPtr</i> | 安全性参数 | SecurityParmsPtr |

相关任务

[使用 MQCONN](#)

MQCNO 概述

可用性: 除 MQCNO_VERSION_4: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: MQCNO 结构允许应用程序指定与本地队列管理器的连接相关的选项。该结构是 MQCONN 调用上的输入/输出参数。有关使用共享句柄和 MQCONN 调用的更多信息, 请参阅 [与 MQCONN 的共享 \(独立于线程的\) 连接](#)。

版本: 为受支持编程语言提供的头, COPY 和 INCLUDE 文件包含最新版本的 MQCNO, 但 *Version* 字段的初始值设置为 MQCNO_VERSION_1。要使用 version-1 结构中不存在的字段, 应用程序必须将 *Version* 字段设置为所需版本的版本号。

字符集和编码: MQCNO 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集, 以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 WebSphere MQ MQI 客户机运行, 那么该结构必须采用客户机的字符集和编码。

MQCNO 的字段

MQCNO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

ClientConn 偏移量 (MQLONG)

ClientConnOffset 是 MQCD 通道定义结构从 MQCNO 结构开始的偏移量 (以字节为单位)。偏移可以是正数或负数。此字段是初始值为 0 的输入字段。

仅当发出 MQCONN 调用的应用程序作为 WebSphere MQ MQI 客户机运行时, 才使用 *ClientConnOffset*。有关如何使用此字段的信息, 请参阅 *ClientConnPtr* 字段的描述。

如果 *Version* 小于 MQCNO_VERSION_2, 那么将忽略此字段。

ClientConnPtr (MQPTR)

ClientConnPtr 是输入字段。其初始值是那些支持指针的编程语言中的空指针, 否则为全空字节字符串。

仅当发出 MQCONN 调用的应用程序作为 WebSphere MQ MQI 客户机运行时, 才使用 *ClientConnOffset* 和 *ClientConnPtr*。通过指定其中一个或其他字段, 应用程序可以通过提供包含所需值的 MQCD 通道定义结构来控制客户机连接通道的定义。

如果应用程序作为 WebSphere MQ MQI 客户机运行, 但未提供 MQCD 结构, 那么将使用 MQSERVER 环境变量来选择通道定义。如果未设置 MQSERVER, 那么将使用客户机通道表。

如果应用程序未作为 WebSphere MQ MQI 客户机运行, 那么将忽略 *ClientConnOffset* 和 *ClientConnPtr*。

如果应用程序提供 MQCD 结构，请将列出的字段设置为必需的值；将忽略 MQCD 中的其他字段。可以用空格填充字符串到字段的长度，或者用空字符终止这些字符串。有关 MQCD 结构中的字段的更多信息，请参阅第 923 页的『字段』。

| MQCD 中的字段 | 值 |
|-------------------------------|---|
| <i>ChannelName</i> | 通道名称。 |
| <i>Version</i> | 结构版本号。不得小于 MQCD_VERSION_7。 |
| <i>TransportType</i> | 任何受支持的传输类型。 |
| <i>ModeName</i> | LU 6.2 方式名。 |
| <i>TpName</i> | LU 6.2 事务程序名。 |
| <i>SecurityExit</i> | 通道安全出口的名称。 |
| <i>SendExit</i> | 通道发送出口的名称。 |
| <i>ReceiveExit</i> | 通道接收出口的名称。 |
| <i>MaxMsgLength</i> | 可以通过客户机连接通道发送的消息的最大长度 (以字节计)。 |
| <i>SecurityUserData</i> | 安全出口的用户数据。 |
| <i>SendUserData</i> | 用于发送出口的用户数据。 |
| <i>ReceiveUserData</i> | 用于接收出口的用户数据。 |
| <i>UserIdentifier</i> | 用于建立 LU 6.2 会话的用户标识。 |
| <i>Password</i> | 用于建立 LU 6.2 会话的密码。 |
| <i>ConnectionName</i> | 连接名称。 |
| <i>HeartbeatInterval</i> | 脉动信号流之间的时间 (以秒为单位)。 |
| <i>StrucLength</i> | MQCD 结构的长度。 |
| <i>ExitNameLength</i> | 由 <i>SendExitPtr</i> 和 <i>ReceiveExitPtr</i> 寻址的出口名称的长度。如果 <i>SendExitPtr</i> 或 <i>ReceiveExitPtr</i> 设置为非空指针的值，那么必须大于零。 |
| <i>ExitDataLength</i> | 由 <i>SendUserDataPtr</i> 和 <i>ReceiveUserDataPtr</i> 寻址的出口数据的长度。如果 <i>SendUserDataPtr</i> 或 <i>ReceiveUserDataPtr</i> 设置为非空指针的值，那么必须大于零。 |
| <i>SendExitsDefined</i> | 由 <i>SendExitPtr</i> 寻址的发送出口数。如果为零，那么 <i>SendExit</i> 和 <i>SendUserData</i> 将提供出口名称和数据。如果大于零，那么 <i>SendExitPtr</i> 和 <i>SendUserDataPtr</i> 将提供出口名称和数据，并且 <i>SendExit</i> 和 <i>SendUserData</i> 必须为空白。 |
| <i>ReceiveExitsDefined</i> | 由 <i>ReceiveExitPtr</i> 寻址的接收出口数。如果为零，那么 <i>ReceiveExit</i> 和 <i>ReceiveUserData</i> 将提供出口名称和数据。如果大于零，那么 <i>ReceiveExitPtr</i> 和 <i>ReceiveUserDataPtr</i> 将提供出口名称和数据，并且 <i>ReceiveExit</i> 和 <i>ReceiveUserData</i> 必须为空白。 |
| <i>SendExitPtr</i> | 首次发送出口的的名称的地址。 |
| <i>SendUserDataPtr</i> | 首次发送出口的数据地址。 |
| <i>ReceiveExitPtr</i> | 第一个接收出口的的名称的地址。 |
| <i>ReceiveUserDataPtr</i> | 首次接收出口的数据的地址。 |
| <i>LongRemoteUserIdLength</i> | 长远程用户标识的长度。 |
| <i>LongRemoteUserIdPtr</i> | 长远程用户标识的地址。 |
| <i>RemoteSecurityId</i> | 远程安全标识。 |

| MQCD 中的字段 | 值 |
|--------------------------|---|
| <i>SSLCipherSpec</i> | SSL CipherSpec。 |
| <i>SSLPeerNamePtr</i> | SSL 对等名称的地址。 |
| <i>SSLPeerNameLength</i> | SSL 对等名称的长度。 |
| <i>KeepAliveInterval</i> | 传递到通信堆栈的值，用于通道的保持活动计时 |
| <i>LocalAddress</i> | 本地通信地址，包括要使用的本地网络适配器的 IP 地址以及要用于传出连接的一系列端口。 |

通过以下两种方法之一提供通道定义结构:

- 通过使用偏移量字段 *ClientConnOffset*

在这种情况下，应用程序必须声明包含后跟通道定义结构 MQCD 的 MQCNO 的复合结构，并将 *ClientConnOffset* 设置为通道定义结构从 MQCNO 开始的偏移量。确保此偏移量正确。*ClientConnPtr* 必须设置为空指针或空字节。

对于不支持指针数据类型或以无法移植到不同环境 (例如，COBOL 编程语言) 的方式实现指针数据类型的编程语言，请使用 *ClientConnOffset*。

对于 Visual Basic 编程语言，称为复合结构 MQCNOCD 在头文件 CMQXB.BAS; 此结构包含后跟 MQCD 结构的 MQCNO 结构。通过调用 MQCNOCD_DEFAULTS 子例程来初始化 MQCNOCD。MQCNOCD 与 MQCONN 调用的 MQCONNAny 变体; 请参阅 MQCONN 调用的描述以获取更多详细信息。

- 通过使用指针字段 *ClientConnPtr*

在这种情况下，应用程序可以单独声明通道定义结构与 MQCNO 结构，并将 *ClientConnPtr* 设置为通道定义结构的地址。将 *ClientConnOffset* 设置为零。

将 *ClientConnPtr* 用于以可移植到不同环境 (例如 C 编程语言) 的方式支持指针数据类型的编程语言。

在 C 编程语言中，可以使用宏变量 MQCD_CLIENT_CONN_DEFAULT 为结构提供比 MQCD_DEFAULT 提供的初始值更适合在 MQCONN 调用上使用的初始值。

无论选择哪种技术，都只能使用 *ClientConnOffset* 和 *ClientConnPtr* 之一; 如果两者都非零，那么调用将失败，原因码为 MQRC_CLIENT_CONN_ERROR。

MQCONN 调用完成后，不会再次引用 MQCD 结构。

如果 *Version* 小于 MQCNO_VERSION_2，那么将忽略此字段。

注: 在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串，初始值为全空字节字符串。

ConnectionId (MQBYTE24)

ConnectionId 是唯一的 24 字节标识，允许 WebSphere MQ 可靠地标识应用程序。应用程序可以将此标识用于 PUT 和 GET 调用中的关联。此输出参数在所有编程语言中具有 24 个空字节的初始值。

队列管理器将唯一标识分配给所有连接，但这些连接是建立的。如果 MQCONN 与 V 5 MQCNO 建立连接，那么应用程序可以从返回的 MQCNO 确定 *ConnectionId*。保证分配的标识在 WebSphere MQ 生成的所有其他标识 (例如 *CorrelId*，*MsgID* 和 *GroupId*) 中唯一。

使用 *ConnectionId* 通过 PCF 命令 "查询连接" 或 MQSC 命令 DISPLAY CONN 来标识长时间运行的工作单元。MQSC 命令 (CONN) 所使用的 *ConnectionId* 派生自此处返回的 *ConnectionId*。PCF "查询和停止连接" 命令可以使用此处返回的 *ConnectionId* 而不进行修改。

通过使用 PCF 命令 "停止连接" 或 MQSC 命令 STOP CONN 指定 *ConnectionId*，可以使用 *ConnectionId* 来强制结束长时间运行的工作单元。有关使用这些命令的更多信息，请参阅 [停止连接](#) 和 [STOP CONN](#)。

如果版本低于 MQCNO_VERSION_5，那么不会返回此字段。

此字段的长度由 MQ_CONNECTION_ID_LENGTH 给出。

ConnTag (MQBYTE128)

ConnTag 是队列管理器在此连接期间与受应用程序影响的资源相关联的标记。每个应用程序或应用程序实例都必须对标记使用不同的值，以便队列管理器可以正确序列化对受影响资源的访问。此字段是输入字段，其初始值为 MQCT_NONE。

请参阅 MQCNO_*_CONN_TAG_* 选项的描述，以获取有关要由不同应用程序使用的值的更多详细信息。当应用程序终止或发出 MQDISC 调用时，该标记将不再有效。

注：在 ASCII 或 EBCDIC 中以 MQ 开头的大写，小写或混合大小写的连接标记值将保留供 IBM 产品使用。请勿使用以这些字母开头的连接标记值。

如果不需要标记，请使用以下特殊值：

MQCT_NONE

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQCT_NONE_ARRAY；此常量具有与 MQCT_NONE 相同的值，但是字符数组而不是字符串。

此字段在连接到 z/OS 队列管理器时使用。在其他环境中，指定值 MQCT_NONE。

此字段的长度由 MQ_CONN_TAG_LENGTH 给出。如果 *Version* 小于 MQCNO_VERSION_3，那么将忽略此字段。

选项 (MQLONG)

控制 MQCONN 操作的选项。

记帐选项

如果 *AccountingConnOverride* 队列管理器属性设置为 MQMON_ENABLED，那么以下选项将控制记帐类型：

MQCNO_ACCOUNTING_MQI_ENABLED

当通过将 *MQIAccounting* 属性设置为 MQMON_OFF 在队列管理器定义中关闭监视数据收集时，设置此标志将启用 MQI 记帐数据收集。

MQCNO_ACCOUNTING_MQI_DISABLED

当通过将 *MQIAccounting* 属性设置为 MQMON_OFF 在队列管理器定义中关闭监视数据收集时，设置此标志将停止 MQI 记帐数据收集。

MQCNO_ACCOUNTING_Q_ENABLED

当通过将 *MQIAccounting* 属性设置为 MQMON_OFF 在队列管理器定义中关闭队列记帐数据收集时，设置此标志将对那些在其队列定义的 *MQIAccounting* 字段中指定队列管理器的队列启用记帐数据收集。

MQCNO_ACCOUNTING_Q_DISABLED

通过将 *MQIAccounting* 属性设置为 MQMON_OFF 来关闭队列管理器定义中的队列记帐数据收集时，设置此标志将关闭那些在其队列定义的 *MQIAccounting* 字段中指定了队列管理器的队列的记帐数据收集。

如果未定义任何这些标志，那么将按照 "队列管理器" 属性中的定义对连接进行记帐。

绑定选项

以下选项控制要使用的 WebSphere MQ 绑定的类型。仅指定下列其中一个选项：

MQCNO_STANDARD_BINDING

应用程序和本地队列管理器代理 (用于管理排队操作的组件) 在单独的执行单元 (通常在单独的进程中) 中运行。此安排维护队列管理器的完整性；即，它保护队列管理器免受错误程序的侵害。

如果队列管理器支持多种绑定类型，并且您设置了 MQCNO_STANDARD_BINDING，那么队列管理器将使用 *qm.ini* 文件 (或等效的 Windows 注册表条目) 中 *Connection* 节中的 *DefaultBindType* 属性来选择实际绑定类型。如果未定义此节，或者无法使用该值或者该值不适用于应用程序，那么队列管理器将选择适当的绑定类型。队列管理器设置在连接选项中使用的实际绑定类型。

在应用程序可能未经过完全测试，或者可能不可靠或不可信的情况下，请使用 MQCNO_STANDARD_BINDING。MQCNO_STANDARD_BINDING 是缺省值。

此选项在所有环境中都受支持。

如果链接到 mqm 库，那么会首先尝试使用缺省绑定类型的标准服务器连接。如果底层服务器库未能装入，那么会改为尝试客户机连接。

- 如果指定了 MQ_CONNECT_TYPE 环境变量，那么可以提供下列其中一个选项来更改 MQCONN 或 MQCONNX 的行为 (如果指定了 MQCNO_STANDARD_BINDING)。(存在例外情况：如果指定了 MQCNO_FASTPATH_BINDING 且 MQ_CONNECT_TYPE 设置为 LOCAL 或 STANDARD 以允许管理员对快速路径连接降级，而不对应用程序进行相关更改：

| 值 | 含义 |
|----------|-------------------------------|
| CLIENT | 仅尝试执行客户机连接。 |
| FASTPATH | 此值在先前发行版中受支持，但现在即使指定此值也会将其忽略。 |
| LOCAL | 仅尝试执行服务器连接。快速路径连接降级为标准服务器连接。 |
| 标准 | 支持兼容先前发行版。此值现在被视为 LOCAL。 |

- 如果在调用 MQCONNX 时未设置 MQ_CONNECT_TYPE 环境变量，那么将尝试使用缺省绑定类型的标准服务器连接。如果服务器库未能装入，那么会尝试客户机连接。

MQCNO_FASTPATH_BINDING

应用程序和本地队列管理器代理程序是同一执行单元的一部分。这与典型的绑定方法相反，在此方法中，应用程序和本地队列管理器代理程序在不同的执行单元中运行。

如果队列管理器不支持此类型的绑定，那么将忽略 MQCNO_FASTPATH_BINDING；将继续处理，就像未指定选项一样。

MQCNO_FASTPATH_BINDING 在多个进程耗用的资源多于应用程序使用的整体资源的情况下具有优势。使用快速路径绑定的应用程序称为可信应用程序。

在决定是否使用快速路径绑定时，请考虑以下要点：

- 使用 MQCNO_FASTPATH_BINDING 选项不会阻止应用程序更改或损坏属于队列管理器的消息和其他数据区。仅在已完全评估这些问题的情况下使用此选项。
- 应用程序不得将异步信号或计时器中断 (例如 sigkill) 与 MQCNO_FASTPATH_BINDING 配合使用。对共享内存段的使用也有限制。
- 应用程序必须使用 MQDISC 调用与队列管理器断开连接。
- 应用程序必须先完成，然后才能使用 endmqm 命令结束队列管理器。
- 在 IBM i 上，作业必须在属于 QMQMADM 组的用户概要文件下运行。另外，程序不得异常停止，否则会出现不可预测的结果。
- 在 UNIX 系统上，mqm 用户标识必须是有效用户标识，mqm 组标识必须是有效组标识。要使应用程序以此方式运行，请配置程序以使其由 mqm 用户标识和 mqm 组标识拥有，然后在程序上设置 setuid 和 setgid 许可权位。

WebSphere MQ 对象权限管理器 (OAM) 仍使用实际用户标识进行权限检查。

- 在 Windows 上，程序必须是 mqm 组的成员。64 位应用程序不支持快速路径绑定。

MQCNO_FASTPATH_BINDING 选项在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux 和 Windows。在 z/OS 上，接受但忽略此选项。

有关使用可信应用程序的含义的更多信息，请参阅 [可信应用程序的限制](#)。

MQCNO_SHARED_BINDING

通过 MQCNO_SHARED_BINDING，应用程序与 local-queue-manager 代理共享一些资源。如果队列管理器不支持此类型的绑定，那么会忽略 MQCNO_SHARED_BINDING。将继续处理，好像未指定此选项一样。

MQCNO_ISOLATED_BINDING

在这种情况下，应用程序进程和本地队列管理器代理程序相互隔离，因为它们不共享资源。如果队列管理器不支持这种绑定，那么会忽略 MQCNO_ISOLATED_BINDING。将继续处理，好像未指定此选项一样。

MQCNO_CLIENT_BINDING

指定此选项以使应用程序仅尝试执行客户机连接。此选项具有以下局限性：

- 在 z/OS 上，使用 MQRC_OPTIONS_ERROR 拒绝 MQCNO_CLIENT_BINDING。
- 如果使用非 MQCNO_STANDARD_BINDING 的任何 MQCNO 绑定选项指定 MQCNO_CLIENT_BINDING，那么会将其拒绝，并返回 MQRC_OPTIONS_ERROR。
- MQCNO_CLIENT_BINDING 不可用于 Java 或 .NET，因为它们有自己的机制来选择绑定类型。
- 如果在调用 MQCONN 时未设置 MQ_CONNECT_TYPE 环境变量，那么将尝试使用缺省绑定类型的标准服务器连接。如果服务器库未能装入，那么会尝试客户机连接。

MQCNO_LOCAL_BINDING

指定此选项以使应用程序尝试执行服务器连接。如果还指定了 MQCNO_FASTPATH_BINDING、MQCNO_ISOLATED_BINDING 或 MQCNO_SHARED_BINDING，那么连接会是此类型的连接，并在此部分中记录。否则，会使用缺省绑定类型尝试执行标准服务器连接。MQCNO_LOCAL_BINDING 具有以下局限性：

- 在 z/OS 上，将忽略 MQCNO_LOCAL_BINDING。
- 如果使用非 MQCNO_STANDARD_BINDING 的任何 MQCNO 重新连接选项指定 MQCNO_LOCAL_BINDING，那么会将其拒绝，并返回 MQRC_OPTIONS_ERROR。
- MQCNO_LOCAL_BINDING 不可用于 Java 或 .NET，因为它们有自己的机制来选择绑定类型。
- 如果在调用 MQCONN 时未设置 MQ_CONNECT_TYPE 环境变量，那么将尝试使用缺省绑定类型的标准服务器连接。如果服务器库未能装入，那么会尝试客户机连接。

在 AIX, HP-UX, Solaris, Linux 和 Windows 上，可以将环境变量 MQ_CONNECT_TYPE 与 *Options* 字段指定的绑定类型配合使用，以控制所使用的绑定类型。如果指定此环境变量，那么它必须具有值 FASTPATH 或 STANDARD；如果它具有其他值，那么将忽略该值。环境变量的值区分大小写；请参阅 [MQCONN 环境变量](#) 以获取更多信息。

环境变量与 *Options* 字段进行交互，如下所示：

- 如果省略环境变量，或者为其提供不受支持的值，那么使用快速路径绑定仅由 *Options* 字段确定。
- 如果为环境变量指定受支持的值，那么仅当同时环境变量和 *Options* 字段指定 fastpath 绑定时，才会使用 fastpath 绑定。

连接标记选项

仅当连接到 z/OS 队列管理器并控制连接标记 *ConnTag* 的使用时，才支持这些选项。只能指定下列其中一个选项：

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

此选项请求在本地队列管理器中独占使用连接标记。如果连接标记已在本地队列管理器中使用，那么 MQCONN 调用将失败，原因码为 MQRC_CONN_TAG_IN_USE。使用本地队列管理器所属的队列共享组中其他位置的连接标记不会影响调用结果。

MQCNNO_SERIALIZE_CONN_TAG_QSG

此选项请求在本地队列管理器所属的队列共享组中独占使用连接标记。如果连接标记已在队列共享组中使用，那么 MQCONN 调用将失败，原因码为 MQRC_CONN_TAG_IN_USE。

MQCNNO_RESTRICT_CONN_TAG_Q_MGR

此选项请求在本地队列管理器中共享使用连接标记。如果连接标记已在本地队列管理器中使用，那么如果发出请求的应用程序正在与该标记的现有用户相同的处理作用域中运行，那么 MQCONN 调用可能会成功。如果不满足此条件，那么 MQCONN 调用将失败，原因码为 MQRC_CONN_TAG_IN_USE。在本地队列管理器所属的队列共享组中的其他位置使用连接标记不会影响调用结果。

- 应用程序必须在同一 MVS 地址空间中运行才能共享连接标记。如果使用连接标记的应用程序是客户机应用程序，那么不允许 MQCNNO_RESTRICT_CONN_TAG_Q_MGR。

MQCNNO_RESTRICT_CONN_TAG_QSG

此选项请求在本地队列管理器所属的队列共享组中共享使用连接标记。如果连接标记已在队列共享组中使用，那么 MQCONN 调用可能成功，前提是发出请求的应用程序正在同一处理作用域中运行，并且作为该标记的现有用户连接到同一队列管理器。

如果不满足这些条件，那么 MQCONN 调用将失败，原因码为 MQRC_CONN_TAG_IN_USE。

- 应用程序必须在同一 MVS 地址空间中运行才能共享连接标记。如果使用连接标记的应用程序是客户机应用程序，那么不允许 MQCNNO_RESTRICT_CONN_TAG_QSG。

如果未指定任何这些选项，那么不会使用 *ConnTag*。如果 *Version* 小于 MQCNNO_VERSION_3，那么这些选项无效。

句柄共享选项

这些选项在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux 和 Windows。它们控制同一进程中不同线程 (并行处理单元) 之间的句柄共享。只能指定下列其中一个选项:

MQCNNO_HANDLE_SHARE_NONE

此选项指示连接和对象句柄只能由导致分配句柄的线程 (即发出 MQCONN, MQCONN 或 MQOPEN 调用的线程) 使用。这些句柄不能由属于同一进程的其他线程使用。

MQCNNO_HANDLE_SHARE_BLOCK

此选项指示由进程的一个线程分配的连接和对象句柄可以由属于同一进程的其他线程使用。但是，一次只有一个线程可以使用任何特定句柄; 即，只允许串行使用句柄。如果一个线程尝试使用另一个线程已使用的句柄，那么调用将阻塞 (等待) 直到该句柄变为可用。

MQCNNO_HANDLE_SHARE_NO_BLOCK

这与 MQCNNO_HANDLE_SHARE_BLOCK 相同，只是如果句柄正由另一个线程使用，那么调用会立即完成 MQCC_FAILED 和 MQRC_CALL_IN_PROGRESS，而不是阻塞，直到句柄变为可用为止。

线程可以有零个或一个非共享句柄:

- 指定 MQCNNO_HANDLE_SHARE_NONE 的每个 MQCONN 或 MQCONN 调用都会在第一个调用上返回新的非共享句柄，而在第二个调用和更高的调用上返回相同的非共享句柄 (假定中间没有 MQDISC 调用)。对于第二次和更高版本的调用，原因码为 MQRC_ALREADY_CONNECTED。
- 指定 MQCNNO_HANDLE_SHARE_BLOCK 或 MQCNNO_HANDLE_SHARE_NO_BLOCK 的每个 MQCONN 调用都会在每个调用上返回新的共享句柄。

对象句柄继承与创建对象句柄的 MQOPEN 调用上指定的连接句柄相同的共享属性。此外，工作单元将继续与用于启动工作单元的连接句柄相同的共享属性; 如果使用共享句柄在一个线程中启动工作单元，那么可以使用同一句柄在另一个线程中更新工作单元。

如果未指定句柄共享选项，那么缺省值由环境确定:

- 在 Microsoft Transaction Server (MTS) 环境中，缺省值与 MQCNNO_HANDLE_SHARE_BLOCK 相同。

- 在其他环境中，缺省值与 MQCNO_HANDLE_SHARE_NONE 相同。

重新连接选项

重新连接选项确定连接是否可重新连接。只有客户机连接可重新连接。

MQCNO_RECONNECT_AS_DEF

重新连接选项解析为其缺省值。如果未设置缺省值，那么此选项的值将解析为 DISABLED。该选项的值将传递到服务器，并且可由 PCF 和 MQSC 查询。

MQCNO_RECONNECT

可以将应用程序重新连接到与 MQCONNX 的 QmgrName 参数值一致的任何队列管理器。仅当客户机应用程序与最初与其建立连接的队列管理器之间没有亲缘关系时，才使用 MQCNO_RECONNECT 选项。该选项的值将传递到服务器，并且可由 PCF 和 MQSC 查询。

MQCNO_RECONNECT_DISABLED

无法重新连接应用程序。该选项的值未传递到服务器。

MQCNO_RECONNECT_Q_MGR

只能将应用程序重新连接到它最初连接的队列管理器。如果可以重新连接客户机，但客户机应用程序与最初与其建立连接的队列管理器之间存在亲缘关系，请使用此值。如果想要客户机自动重新连接至高可用性队列管理器的备用实例，那么选择此值。该选项的值将传递到服务器，并且可由 PCF 和 MQSC 查询。

仅将 MQCNO_RECONNECT，MQCNO_RECONNECT_DISABLED 和 MQCNO_RECONNECT_DISABLED 选项用于客户机连接。如果选项用于绑定连接，那么 MQCONNX 将失败，完成代码为 MQCC_FAILED，原因码为 MQRC_OPTIONS_ERROR。WebSphere MQ Java 类不支持自动客户机重新连接

对话共享选项

以下选项仅适用于 TCP/IP 客户机连接。对于 SNA，SPX 和 NetBios 通道，将忽略这些值，并且通道将像先前版本的产品一样运行

MQCNO_NO_CONV_SHARING

此选项不允许对话共享。

您可以在大量装入对话的情况下使用 MQCNO_NO_CONV_SHARING，因此，在存在共享对话的通道实例的服务器连接端上可能存在争用。当连接到支持对话共享的通道时，MQCNO_NO_CONV_SHARING 的行为类似于 sharecnv (1)，当连接到不支持对话共享的通道时，MQCNO_NO_CONV_SHARING 的行为类似于 sharecnv (0)。

MQCNO_ALL_CONVS_SHARE

此选项允许对话共享；应用程序不会对通道实例上的连接数施加任何限制。此选项是缺省值。

如果应用程序指示通道实例可以共享，但通道的服务器连接端上的 *SharingConversations* (SHARECNV) 定义设置为 1，那么不会发生共享，并且不会向应用程序发出警告。

同样，如果应用程序指示允许共享，但服务器连接 *SharingConversations* 定义设置为零，那么不会发出任何警告，并且应用程序在版本低于 7.0 的产品中表现出与客户机相同的行为；将忽略与共享对话相关的应用程序设置。

MQCNO_NO_CONV_SHARING 和 MQCNO_ALL_CONVS_SHARE 互斥。如果在特定连接上指定了这两个选项，那么将拒绝该连接，原因码为 MQRC_OPTIONS_ERROR。

通道定义选项

以下选项控制在 MQCNO 中传递的通道定义结构的使用:

MQCNO_CD_FOR_OUTPUT_ONLY

此选项允许 MQCNO 中的通道定义结构仅用于返回成功 MQCONN 调用上使用的通道名称。

如果未提供有效的通道定义结构, 那么调用将失败, 原因码为 MQRC_CD_ERROR。

如果应用程序未作为客户机运行, 那么将忽略该选项。

可使用 MQCNO_USE_CD_SELECTION 选项在后续 MQCONN 调用上使用返回的通道名称, 以使用相同的通道定义重新连接。当客户机通道表中有多个适用的通道定义时, 这可能很有用。

MQCNO_USE_CD_SELECTION

此选项允许 MQCONN 调用使用 MQCNO 中传递的通道定义结构中包含的通道名称进行连接。

如果设置了 MQSERVER 环境变量, 那么将使用由其定义的通道定义。如果未设置 MQSERVER, 那么将使用客户机通道表。

如果找不到具有匹配通道名称和队列管理器名称的通道定义, 那么调用将失败, 原因码为 MQRC_Q_MGR_NAME_ERROR。

如果未提供有效的通道定义结构, 那么调用将失败, 原因码为 MQRC_CD_ERROR。

如果应用程序未作为客户机运行, 那么将忽略该选项。

缺省选项

如果您不需要上述任何选项, 那么可以使用以下选项:

MQCNO_NONE

未指定任何选项。

使用 MQCNO_NONE 来帮助程序文档。不打算将此选项与任何其他 MQCNO_* 选项一起使用, 但由于其值为零, 因此无法检测到此类使用。

SecurityParms 偏移量 (MQLONG)

SecurityParmsOffset 是 MQCSP 结构从 MQCNO 结构开始的偏移量 (以字节为单位)。偏移可以是正数或负数。此字段是输入字段, 初始值为 0。

如果 *V* 小于 MQCNO_VERSION_5, 那么将忽略此字段。

MQCSP 结构在 [第 296 页的『MQCSP-安全性参数』](#)中定义。

SecurityParmsPtr (PMQCSP)

SecurityParmsPtr 是 MQCSP 结构的地址, 用于指定用户标识和密码以供授权服务进行认证。此字段是输入字段, 其初始值为空指针或空字节。

如果 *V* 小于 MQCNO_VERSION_5, 那么将忽略此字段。

MQCSP 结构在 [第 296 页的『MQCSP-安全性参数』](#)中定义。

SSLConfigOffset (MQLONG)

SSLConfigOffset 是 MQSCO 结构从 MQCNO 结构开始的偏移量 (以字节为单位)。偏移可以是正数或负数。此字段是输入字段, 初始值为 0。

仅当发出 MQCONN 调用的应用程序作为 WebSphere MQ MQI 客户机运行时, 才使用 *SSLConfigOffset*。有关如何使用此字段的信息, 请参阅 *SSLConfigPtr* 字段的描述。

如果 *Version* 小于 MQCNO_VERSION_4, 那么将忽略此字段。

SSLConfigPtr (PMQSCO)

SSLConfigPtr 是输入字段。Itsinitial 值是那些支持指针的编程语言中的空指针, 否则为全空字节字符串。

仅当发出 MQCONN 调用的应用程序作为 WebSphere MQ MQI 客户机运行并且通道协议为 TCP/IP 时，才使用 *SSLConfigPtr* 和 *SSLConfigOffset*。如果应用程序未作为 WebSphere MQ 客户机运行，或者通道协议不是 TCP/IP，那么将忽略 *SSLConfigPtr* 和 *SSLConfigOffset*。

通过指定 *SSLConfigPtr* 或 *SSLConfigOffset* 以及 *ClientConnPtr* 或 *ClientConnOffset*，应用程序可以控制对客户机连接使用 SSL。以此方式指定 SSL 信息时，将忽略环境变量 MQSSLKEYR 和 MQSSLCRYP；还将忽略客户机通道定义表 (CCDT) 中的任何与 SSL 相关的信息。

只能在以下位置指定 SSL 信息：

- 客户机进程的第一个 MQCONN 调用，或者
- 在使用 MQDISC 结束与队列管理器的所有先前 SSL/TLS 连接时，后续 MQCONN 调用。

这些是可以初始化进程范围 SSL 环境的唯一状态。如果在 SSL 环境已存在时发出 MQCONN 调用并指定 SSL 信息，那么将忽略调用上的 SSL 信息，并使用现有 SSL 环境进行连接；在这种情况下，调用将返回完成代码 MQCC_WARNING 和原因码 MQRC_SSL_ALREADY_初始化。

您可以通过在 *SSLConfigPtr* 中指定地址或在 *SSLConfigOffset* 中指定偏移量，以与 MQCD 结构相同的方式提供 MQSCO 结构；请参阅 *ClientConnPtr* 的描述以获取有关如何执行此操作的详细信息。但是，只能使用 *SSLConfigPtr* 和 *SSLConfigOffset* 中的一个；调用失败，原因码为 MQRC_SSL_CONFIG_ERROR。如果两者都是非零的。

MQCONN 调用完成后，不会再次引用 MQSCO 结构。

如果 *Version* 小于 MQCNO_VERSION_4，那么将忽略此字段。

注：在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串。

StrucId (MQCHAR4)

StrucId 始终是输入字段。其初始值为 MQCNO_STRUC_ID。

该值必须为：

MQCNO_STRUC_ID

连接选项结构的标识。

对于 C 编程语言，还定义了常量 MQCNO_STRUC_ID_ARRAY；此常量具有与 MQCNO_STRUC_ID 相同的值，但是字符数组而不是字符串。

Version (MQLONG)

版本始终是输入字段。其初始值为 MQCNO_VERSION_1。

值必须为以下其中一项：

MQCNO_VERSION_1

Version-1 连接选项结构。

MQCNO_VERSION_2

Version-2 连接选项结构。

MQCNO_VERSION_3

Version-3 连接选项结构。

MQCNO_VERSION_4

Version-4 连接选项结构。

MQCNO_VERSION_5

Version-5 连接选项结构。

此版本的 MQCNO 结构在 z/OS 上扩展了 MQCNO_VERSION_3，在所有其他平台上扩展了 MQCNO_VERSION_4。

仅在结构的最新版本中存在的字段在字段的描述中标识为此类字段。以下常量指定当前版本的版本号：

MQCNO_CURRENT_VERSION

当前版本的连接选项结构。

MQCNO 的初始值和语言声明

表 488: MQCNO 的 MQCNO 中字段的初始值

| 字段名称 | 常量的名称 | 常量值 |
|----------------------------|-----------------|---------|
| <i>StrucId</i> | MQCNO_STRUC_ID | 'CNO~' |
| <i>Version</i> | MQCNO_VERSION_1 | 1 |
| <i>Options</i> | MQCNO_NONE | 0 |
| <i>ClientConnOffset</i> | None | 0 |
| <i>ClientConnPtr</i> | None | 空指针或空字节 |
| <i>ConnTag</i> | MQCT_NONE | Null |
| <i>SSLConfigPtr</i> | None | 空指针或空字节 |
| <i>SSLConfigOffset</i> | None | 0 |
| <i>ConnectionId</i> | None | 空指针或空字节 |
| <i>SecurityParmsOffset</i> | None | 空指针或空字节 |
| <i>SecurityParmsPtr</i> | None | 空指针或空字节 |

注意:

1. 符号 ~ 表示单个空白字符。
2. 在 C 编程语言中，宏变量 MQCNO_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQCNO Mycno = {MQCNO_DEFAULT};
```

C 声明

```
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 声明

```
** 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.

```

PL/I 声明

```

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 MQCONN */
  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 */

```

High Level Assembler 声明

```

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
*               MQCONN
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
MQCNO_AREA       DS   CL(MQCNO_LENGTH)

```

Visual Basic 声明

```

Type MQCNO
  StrucId          As String*4 'Structure identifier'
  Version          As Long     'Structure version number'
  Options          As Long     'Options that control the action of
                              'MQCONN'
  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-安全性参数

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|---------------------------|---------------------|------------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>AuthenticationType</i> | 认证类型 | AuthenticationType |
| <i>Reserved1</i> | 对于 IBM i 上的指针对齐是必需的 | Reserved1 |
| <i>CSPUserIdPtr</i> | 用户标识的地址 | CSPUserIdPtr |
| <i>CSPUserIdOffset</i> | 用户标识的偏移量 | CSPUserId 偏移量 |
| <i>CSPUserIdLength</i> | 用户标识的长度 | CSPUserId 长度 |
| <i>Reserved2</i> | 对于 IBM i 上的指针对齐是必需的 | Reserved2 |
| <i>CSPPasswordPtr</i> | 密码的地址 | CSPPasswordPtr |
| <i>CSPPasswordOffset</i> | 密码的偏移量 | CSPPasswordOffset |
| <i>CSPPasswordLength</i> | 密码的长度 | CSPPasswordLength |

MQCSP 概述

可用性: 所有 WebSphere MQ 产品。

用途: MQCSP 结构使授权服务能够认证用户标识和密码。在 MQCONN 调用上指定 MQCSP 连接安全性参数结构。

字符集和编码: MQCSP 中的数据必须包含在本地队列管理器的字符集和编码中; 这些数据分别由 *CodedCharSetId* 队列管理器属性和 MQENC_NATIVE 提供。

MQCSP 的字段

MQCSP 结构包含以下字段; 这些字段按字母顺序进行描述:

AuthenticationType (MQLONG)

AuthenticationType 是输入字段。其初始值为 MQCSP_AUTH_NONE。

这是要执行的认证类型。有效值包括:

MQCSP_AUTH_NONE

请勿使用用户标识和密码字段。

MQCSP_AUTH_USER_ID_AND_PWD

认证用户标识和密码字段。

CSPPasswordLength (MQLONG)

此字段是要在认证中使用的密码的长度。

密码的最大长度取决于平台，请参阅 [用户标识](#)。如果密码长度大于允许的最大长度，那么认证请求将失败并返回 MQRC_NOT_AUTHORIZED。

此字段是输入字段。此字段的初始值为 0。

CSPPasswordOffset (MQLONG)

这是要用于认证的密码的偏移量 (以字节计)。偏移可以是正数或负数。

这是一个输入字段。此字段的初始值为 0。

CSPPasswordPtr (MQPTR)

这是要用于认证的密码的地址 (以字节计)。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针，否则为全空字节字符串。如果 *Version* 小于 MQCNO_VERSION_5，那么将忽略此字段。

CSPUserId 长度 (MQLONG)

此字段是要在认证中使用的用户标识的长度。

用户标识的最大长度取决于平台，请参阅 [用户标识](#)。如果用户标识的长度大于允许的最大长度，那么认证请求将失败并返回 MQRC_NOT_AUTHORIZED。

此字段是输入字段。此字段的初始值为 0。

CSPUserId 偏移量 (MQLONG)

这是要用于认证的用户标识的偏移量 (以字节计)。偏移可以是正数或负数。

这是一个输入字段。此字段的初始值为 0。

CSPUserIdPtr (MQPTR)

这是要在认证中使用的用户标识的地址 (以字节计)。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针，否则为全空字节字符串。如果 *Version* 小于 MQCNO_VERSION_5，那么将忽略此字段。

Reserved1 (MQBYTE4)

保留字段，对于 IBM i 上的指针对齐是必需的。

这是一个输入字段。此字段的初始值全部为空。

Reserved2 (MQBYTE8)

保留字段，对于 IBM i 上的指针对齐是必需的。

这是一个输入字段。此字段的初始值全部为空。

StrucId (MQCHAR4)

结构标识。

该值必须为:

MQCSP_STRUC_ID

安全性参数结构的标识。

对于 C 编程语言，还定义了常量 MQCSP_STRUC_ID_ARRAY; 此值与 MQCSP_STRUC_ID 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQCSPSTRUC_ID。

Version (MQLONG)

结构版本号。

该值必须为:

MQCSP_VERSION_1

Version-1 安全性参数结构。

以下常量指定当前版本的版本号:

MQCSP_CURRENT_VERSION

当前版本的安全参数结构。

这始终是一个输入字段。此字段的初始值为 MQCSP_VERSION_1。

MQCSP 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|--------------------|-----------------|-----------------|
| StrucId | MQCSP_STRUC_ID | 'CSP↵' |
| Version | MQCSP_VERSION_1 | 1 |
| AuthenticationType | None | MQCSP_AUTH_NONE |
| Reserved1 | None | 空字符串或空白 |
| CSPUserIdPtr | None | 空指针或空字节 |
| CSPUserIdOffset | None | 0 |
| CSPUserIdLength | None | 0 |
| Reserved2 | None | 空字符串或空白 |
| CSPPasswordPtr | None | 空指针或空字节 |
| CSPPasswordOffset | None | 0 |
| CSPPasswordLength | None | 0 |

注意:

1. 符号 ↵ 表示单个空白字符。
2. 在 C 编程语言中, 宏变量 MQCSP_DEFAULT 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值:

```
MQCSP MyCSP = {MQCSP_DEFAULT};
```

C 声明

```
typedef struct tagMQCSP MQCSP;  
struct tagMQCSP {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQLONG     AuthenticationType; /* Type of authentication */  
    MQBYTE4    Reserved1;       /* Required for IBM i pointer  
                                alignment */  
    MQPTR      CSPUserIdPtr;     /* Address of user ID */  
    MQLONG     CSPUserIdOffset;  /* Offset of user ID */  
    MQLONG     CSPUserIdLength;  /* Length of user ID */  
    MQBYTE8    Reserved2;       /* Required for IBM i pointer  
                                alignment */  
    MQPTR      CSPPasswordPtr;   /* Address of password */  
    MQLONG     CSPPasswordOffset; /* Offset of password */
```

```

MQLONG      CSPPasswordLength; /* Length of password */
};

```

COBOL 声明

```

** 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.

```

PL/I 声明

```

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 */

```

Visual Basic 声明

```

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-控制回调选项结构

下表汇总了结构中的字段。指定控制回调函数的结构。

| 表 491: MQCTLO 中的字段 | | |
|-----------------------|-------------|--------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucID</i> | 结构标识 | StrucID |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 选项 | 选项 |
| <i>Reserved</i> | 保留字段 | 选项 |
| <i>ConnectionArea</i> | 要使用的回调函数的字段 | ConnectionArea |

MQCTLO 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS 以及连接到这些系统的 WebSphere MQ MQI 客户机。MQCTLO 结构概述。

用途:MQCTLO 结构用于指定与控制回调函数相关的选项。

此结构是 [第 594 页](#) 的『MQCTL-控制回调』调用上的输入和输出参数。

版本:MQCTLO 的当前版本为 MQCTLO_VERSION_1。

字符集和编码:MQCTLO 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集, 以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 MQ MQI 客户机运行, 那么该结构必须采用客户机的字符集和编码。

MQCTLO 的字段

MQCTLO 结构的字段的字母列表。

MQCTLO 结构包含以下字段; 这些字段按字母顺序描述:

ConnectionArea (MQPTR)

控制选项结构- *ConnectionArea* 字段

这是可供回调函数使用的字段。

队列管理器不会根据此字段的内容做出任何决策, 并且会将其按原样传递到 MQCBC 结构中的 [第 253 页](#) 的『*ConnectionArea* (MQPTR)』字段 (这是回调的输入参数)。

对于除 MQOP_START 和 MQOP_START_WAIT 以外的所有操作, 将忽略此字段。

这是回调函数的输入和输出字段。此字段的初始值为空指针或空字节。

选项 (MQLONG)

控制选项结构-"选项" 字段

用于控制 MQCTL 操作的选项。

MQCTLO_FAIL_IF QUIESCING

如果队列管理器或连接处于停顿状态, 那么强制 MQCTL 调用失败。

在 MQCB 调用上传递的 MQGMO 选项中指定 MQGMO_FAIL_IF QUIESCING, 以在消息使用者停顿向其发出通知。

MQCTLO_THREAD_AFFINITY

此选项通知系统应用程序要求在同一线程上调用同一连接的所有消息使用者。此线程将用于使用者的所有调用, 直到连接停止为止。

缺省选项: 如果不需要任何描述的选项, 请使用以下选项:

MQCTLO_NONE

使用此值来指示未指定任何其他选项; 所有选项均采用其缺省值。MQCTLO_NONE 定义为帮助程序文档; 不打算将此选项与任何其他选项一起使用, 但由于其值为零, 因此无法检测到此类使用。

这是一个输入字段。Options 字段的初始值为 MQCTLO_NONE。

保留 (MQLONG)

这是保留字段。该值必须为零。

StrucId (MQCHAR4)

控制选项结构- *StrucId* 字段

这是结构标识; 值必须为:

MQCTLO_STRUC_ID

"控制选项" 结构的标识。

对于 C 编程语言, 还定义了常量 MQCTLO_STRUC_ID_ARRAY; 此值与 MQCTLO_STRUC_ID 相同, 但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQCTLO_STRUC_ID。

Version (MQLONG)

控制选项结构-"版本" 字段

这是结构版本号; 值必须为:

MQCTLO_VERSION_1

Version-1 控制选项结构。

以下常量指定当前版本的版本号:

MQCTLO_CURRENT_VERSION

控制选项结构的当前版本。

这始终是一个输入字段。此字段的初始值为 MQCTLO_VERSION_1。

MQCTLO 的初始值和语言声明

控制选项结构-初始值

| 表 492: MQCTLO 中字段的初始值 | | |
|---|------------------|---------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQCTLO_STRUC_ID | 'CTLO' |
| <i>Version</i> | MQCTLO_VERSION_1 | 1 |
| <i>Options</i> | MQCTLO_NONE | Null |
| <i>Reserved</i> | 保留字段 | |
| <i>ConnectionArea</i> | None | 空指针或空字节 |
| <p>注意:</p> <p>1. 在 C 编程语言中, 宏变量 MQCTLO_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:</p> <pre style="background-color: #f0f0f0; padding: 10px;">MQCTLO MyCTLO = {MQCTLO_DEFAULT};</pre> | | |

C 声明

控制选项结构-C 语言声明

```
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 声明

```
** 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
```

PL/I 声明

```
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 - 分发头

下表汇总了结构中的字段。

| 表 493: MQDH 中的字段 | | |
|------------------------|---------------------------|---------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>StrucLength</i> | MQDH 结构及以下记录的长度 | StrucLength |
| <i>Encoding</i> | MQPMR 记录数组之后数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | MQPMR 记录数组之后数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | MQPMR 记录数组之后数据的格式名称 | FORMAT |
| <i>Flags</i> | 常规标志 | 标志 |
| <i>PutMsgRecFields</i> | 指示哪些 MQPMR 字段存在的标志 | PutMsgRecFields |
| <i>RecsPresent</i> | 存在的对象记录数 | RecsPresent |
| <i>ObjectRecOffset</i> | 第一个对象记录与 MQDH 开始处之间的偏移量 | ObjectRec 偏移量 |
| <i>PutMsgRecOffset</i> | 第一个放置消息记录与 MQDH 开始处之间的偏移量 | PutMsgRecOffset |

MQDH 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

用途:MQDH 结构描述了当消息是存储在传输队列上的分发列表消息时, 该消息中存在的其他数据。分发列表消息是发送到多个目标队列的消息。附加数据包含 MQDH 结构, 后跟 MQOR 记录数组和 MQPMR 记录数组。

此结构由专门的应用程序使用, 这些应用程序将消息直接放在传输队列上, 或者从传输队列中除去消息 (例如: 消息通道代理程序)。

要将消息放入分发列表的应用程序不得使用此结构。相反, 它们必须使用 MQOD 结构来定义分发列表中的目标, 并使用 MQPMO 结构来指定消息属性或接收有关发送到各个目标的消息的信息。

格式名:MQFMT_DIST_HEADER。

字符集和编码:MQDH 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。

将 MQDH 的字符集和编码设置为 *CodedCharSetId* 和 *Encoding* 字段:

- MQMD (如果 MQDH 结构位于消息数据的开头), 或者
- MQDH 结构之前的头结构 (所有其他情况)。

用法:当应用程序将消息放入分发列表, 并且部分或所有目标都是远程目标时, 队列管理器会使用 MQXQH 和 MQDH 结构作为应用程序消息数据的前缀, 并将消息放在相关传输队列上。因此, 当消息位于传输队列中时, 数据按以下顺序出现:

- MQXQH 结构
- MQDH 结构以及 MQOR 和 MQPMR 记录的数组
- 应用程序消息数据

根据目标, 队列管理器可以生成多条这样的消息, 并将其放置在不同的传输队列上。在这种情况下, 这些消息中的 MQDH 结构标识由应用程序打开的分发列表定义的目标的不同子集。

将分发列表消息直接放在传输队列上的应用程序必须符合上述顺序, 并且必须确保 MQDH 结构正确。如果 MQDH 结构无效, 那么队列管理器可能会使 MQPUT 或 MQPUT1 调用失败, 原因码为 MQRC_DH_ERROR。

仅当已将队列定义为能够支持分发列表消息时, 才能将消息以分发列表形式存储在队列上 (请参阅第 731 页的『队列的属性』中描述的 *DistLists* 队列属性)。如果应用程序将分发列表消息直接放置在不支持分发列表的队列上, 那么队列管理器会将分发列表消息分割为单独的消息, 并改为将这些消息放置在队列上。

MQDH 的字段

MQDH 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

CodedCharSetId (MQLONG)

这是 MQOR 和 MQPMR 记录数组后面的数据的字符集标识; 它不适用于 MQDH 结构本身中的字符数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。可以使用以下特殊值:

MQCCSI_INHERIT

继承此结构的字符集标识。

遵循 此结构的数据中的字符数据与此结构位于同一字符集中。

队列管理器将消息中发送的结构中的此值更改为结构的实际字符集标识。如果未发生错误, 那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER, 那么不能使用 MQCCSI_INHERIT。

此值在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

此字段的初始值为 MQCCSI_UNDEFINED。

Encoding (MQLONG)

这是遵循 MQOR 和 MQPMR 记录数组的数据的数字编码; 它不适用于 MQDH 结构本身中的数字数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。

此字段的初始值为 0。

Flags (MQLONG)

可以指定以下标志:

MQDHF_NEW_MSG_IDS

为分发列表中的每个目标生成新的消息标识。仅当不存在任何放置消息记录时,或者当存在记录但这些记录不包含 *MsgId* 字段时,才设置此值。

使用此标志将延迟生成消息标识,直到最终将分发列表消息拆分为各个消息为止。这将最小化必须随分发列表消息一起流动的控制信息量。

当应用程序将消息放入分发列表时,队列管理器会在 MQDH 中设置 MQDHF_NEW_MSG_IDS,当以下两种情况均为 true 时,会生成 MQDH 中的 MQDHF_NEW_MSG_IDS:

- 应用程序未提供 *put-message* 记录,或者提供的记录不包含 *MsgId* 字段。
- MQMD 中的 *MsgId* 字段为 MQMI_NONE,或者 MQPMO 中的 *Options* 字段包含 MQPMO_NEW_MSG_ID

如果不需要任何标志,请指定以下内容:

MQDHF_NONE

未指定任何标志。定义 MQDHF_NONE 以帮助程序文档。不打算将此常量与任何其他常量一起使用,但由于其值为零,因此无法检测到此类使用。

此字段的初始值为 MQDHF_NONE。

Format (MQCHAR8)

这是遵循 MQOD 和 MQPMR 记录数组的数据的格式名称(以最后出现者为准)。

在 MQPUT 或 MQPUT1 调用上,应用程序必须将此字段设置为适合于数据的值。此字段的编码规则与 MQMD 中 *Format* 字段的编码规则相同。

此字段的初始值为 MQFMT_NONE。

ObjectRec 偏移量 (MQLONG)

这将提供包含目标队列名称的 MQOR 对象记录数组中第一条记录的偏移量(以字节计)。此数组中有 *RecsPresent* 条记录。这些记录(加上第一个对象记录和前一个字段之间跳过的任何字节)包含在 *StrucLength* 字段给出的长度中。

分发列表必须始终至少包含一个目标,因此 *ObjectRecOffset* 必须始终大于零。

此字段的初始值为 0。

PutMsgRecFields (MQLONG)

您可以不指定或不指定以下任何标志:

MQPMRF_MSG_ID

存在消息标识字段。

MQPMRF_CORREL_ID

存在相关标识字段。

MQPMRF_GROUP_ID

存在组标识字段。

MQPMRF_FEEDBACK

存在反馈字段。

MQPMRF_ACCOUNTING_TOKEN

存在记帐标记字段。

如果不存在 MQPMR 字段,请指定以下内容:

MQPMRF_NONE

不存在放置消息记录字段。定义 MQPMRF_NONE 以帮助程序文档。不打算将此常量与任何其他常量一起使用，但由于其值为零，因此无法检测到此类使用。

此字段的初始值为 MQPMRF_NONE。

PutMsgRecOffset (MQLONG)

这将提供包含消息属性的 MQPMR 放入消息记录数组中第一条记录的偏移量 (以字节计)。如果存在，那么此数组中有 *RecsPresent* 条记录。这些记录 (加上第一个放入消息记录和前一个字段之间跳过的任何字节) 包含在 *StrucLength* 字段给出的长度中。

放置消息记录是可选的; 如果未提供任何记录，那么 *PutMsgRecOffset* 为零，并且 *PutMsgRecFields* 的值为 MQPMRF_NONE。

此字段的初始值为 0。

RecsPresent (MQLONG)

这是目标数。分发列表必须始终至少包含一个目标，因此 *RecsPresent* 必须始终大于零。

此字段的初始值为 0。

StrucId (MQCHAR4)

该值必须为:

MQDH_STRUC_ID

分发头结构的标识。

对于 C 编程语言，还定义了常量 MQDH_STRUC_ID_ARRAY; 此值与 MQDH_STRUC_ID 相同，但是字符数组而不是字符串。

此字段的初始值为 MQDH_STRUC_ID。

StrucLength (MQLONG)

这是从 MQDH 结构开始到消息数据开始跟随 MQOR 和 MQPMR 记录数组的字节数。数据按以下顺序出现:

- MQDH 结构
- MQOR 记录数组
- MQPMR 记录数组
- 消息数据

MQOR 和 MQPMR 记录的数组由 MQDH 结构中包含的偏移量寻址。如果这些偏移导致一个或多个 MQDH 结构，记录数组和消息数据之间存在未使用的字节，那么这些未使用的字节必须包含在 *StrucLength* 的值中，但队列管理器不会保留这些字节的内容。它对 MQPMR 记录数组之前的 MQOR 记录数组有效。

此字段的初始值为 0。

Version (MQLONG)

该值必须为:

MQDH_VERSION_1

分发头结构的版本号。

以下常量指定当前版本的版本号:

MQDH_CURRENT_VERSION

分发头结构的当前版本。

此字段的初始值为 MQDH_VERSION_1。

MQDH 的初始值和语言声明

表 494: MQDH 的 MQDH 中字段的初始值

| 字段名称 | 常量的名称 | 常量值 |
|------------------------|------------------|--------|
| <i>StrucId</i> | MQDH_STRUC_ID | 'DH~~' |
| <i>Version</i> | MQDH_VERSION_1 | 1 |
| <i>StrucLength</i> | None | 0 |
| <i>Encoding</i> | None | 0 |
| <i>CodedCharSetId</i> | MQCCSI_UNDEFINED | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQDHF_NONE | 0 |
| <i>PutMsgRecFields</i> | MQPMRF_NONE | 0 |
| <i>RecsPresent</i> | None | 0 |
| <i>ObjectRecOffset</i> | None | 0 |
| <i>PutMsgRecOffset</i> | None | 0 |

注意:

1. 符号 ~ 表示单个空白字符。
2. 在 C 编程语言中，宏变量 MQDH_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQDH MyDH = {MQDH_DEFAULT};
```

C 声明

```
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 声明

```
** 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-STRUCLNGTH  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.

```

PL/I 声明

```

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 */

```

Visual Basic 声明

```

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-死信头

下表汇总了结构中的字段。

表 495: MQDLH 中的字段

| 字段 | 描述 | Topic |
|-----------------------|---------------------|--------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Reason</i> | 到达死信队列的原因消息 | 原因 |
| <i>DestQName</i> | 原始目标队列的名称 | DestQName |
| <i>DestQMGrName</i> | 原始目标队列管理器的名称 | DestQMGrName |
| <i>Encoding</i> | MQDLH 之后数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | MQDLH 之后数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | MQDLH 之后数据的格式名称 | FORMAT |
| <i>PutApplType</i> | 将消息放置在死信队列上的应用程序的类型 | PutApplType |
| <i>PutApplName</i> | 将消息放置在死信队列上的应用程序的名称 | PutApplName |
| <i>PutDate</i> | 消息放置在死信队列上的日期 | PutDate |
| <i>PutTime</i> | 消息放置在死信队列上的时间 | PutTime |

MQDLH 概述

可用性: 所有 WebSphere MQ 平台。

用途: MQDLH 结构描述以死信 (undelivered-message) 队列上的消息的应用程序消息数据为前缀的信息。消息可以到达死信队列，原因是队列管理器或消息通道代理已将其重定向到队列，或者应用程序已将消息直接放入队列。

格式名: MQFMT_DEAD_LETTER_HEADER。

字符集和编码: MQDLH 结构中的字段是由 *CodedCharSetId* 和 *Encoding* 字段提供的字符集和编码。这些在 MQDLH 之前的头结构中指定，或者在 MQMD 结构中指定 (如果 MQDLH 位于应用程序消息数据的开头)。

字符集必须是对队列名称中有效的字符具有单字节字符的字符集。

如果您正在使用 WMQ classes for Java/JMS，并且 Java 虚拟机不支持 MQMD 中定义的代码页，那么将以 UTF-8 字符集编写 MQDLH。

用法: 将消息直接放在死信队列上的应用程序必须以 MQDLH 结构作为消息数据的前缀，并使用相应的值初始化字段。但是，队列管理器不要求 MQDLH 结构存在，也不要求为字段指定有效值。

如果消息太长而无法放入死信队列，那么应用程序必须执行下列其中一项操作：

- 截断消息数据以适合死信队列。
- 将该消息记录在辅助存储器上，并将异常报告消息放置在死信队列上以指示此情况。
- 废弃消息并向其发起方返回错误。如果消息是 (或可能是) 关键消息，那么仅当已知发起方仍具有消息副本时才执行此操作；例如，消息通道代理程序从通信通道接收的消息。

以上哪一项是合适的 (如果有的话) 取决于应用程序的设计。

当作为段的消息在前面放置 MQDLH 结构时，队列管理器将执行特殊处理；请参阅 MQMDE 结构的描述以获取更多详细信息。

将消息放入死信队列: 将消息放入死信队列时，用于 MQPUT 或 MQPUT1 调用的 MQMD 结构必须与与消息关联的 MQMD (通常是 MQGET 调用返回的 MQMD) 相同，但以下内容除外：

- 将 *CodedCharSetId* 和 *Encoding* 字段设置为用于 MQDLH 结构中的字段的任何字符集和编码。
- 将 *Format* 字段设置为 MQFMT_DEAD_LETTER_HEADER，以指示数据以 MQDLH 结构开头。

- 使用适合于以下情况的上下文选项来设置上下文字段 (*AccountingToken*, *ApplIdentityData*, *ApplOriginData*, *PutApplName*, *PutApplType*, *PutDate*, *PutTime* 和 *UserIdentifier*):
 - 将不与任何先前消息相关的消息放入死信队列的应用程序必须使用 MQPMO_DEFAULT_CONTEXT 选项; 这会导致队列管理器将消息描述符中的所有上下文字段设置为其缺省值。
 - 将刚收到的消息放入死信队列的服务器应用程序必须使用 MQPMO_PASS_ALL_CONTEXT 选项来保留原始上下文信息。
 - 将 应答 放入死信队列的服务器应用程序必须使用 MQPMO_PASS_IDENTITY_CONTEXT 选项; 这将保留身份信息, 但将源信息设置为服务器应用程序的源信息。
 - 将消息从其通信通道接收到的消息放入死信队列的消息通道代理必须使用 MQPMO_SET_ALL_CONTEXT 选项来保留原始上下文信息。

在 MQDLH 结构本身中, 按如下所示设置字段:

- 将 *CodedCharSetId*, *Encoding* 和 *Format* 字段设置为描述 MQDLH 结构后的数据的值, 通常是原始消息描述符中的值。
- 将上下文字段 *PutApplType*, *PutApplName*, *PutDate* 和 *PutTime* 设置为与将消息放入死信队列的应用程序相应的值; 这些值与原始消息无关。
- 根据需要设置其他字段。

确保所有字段都具有有效值, 并且字符字段用空白填充到定义的字段长度; 不要使用空字符过早结束字符数据, 因为队列管理器不会将空字符和后续字符转换为 MQDLH 结构中的空白。

从死信队列获取消息: 从死信队列获取消息的应用程序必须验证消息是否以 MQDLH 结构开头。应用程序可以通过检查消息描述符 MQMD 中的 *Format* 字段来确定是否存在 MQDLH 结构; 如果该字段的值为 MQFMT_DEAD_LETTER_HEADER, 那么消息数据以 MQDLH 结构开头。另请注意, 如果应用程序从死信队列获取的消息最初对于该队列太长, 那么可能会截断这些消息。

MQDLH 的字段

MQDLH 结构包含以下字段; 这些字段按字母顺序进行描述:

CodedCharSetId (MQLONG)

CodedCharSetId 是流经 MQDLH 结构的数据 (通常是来自原始消息的数据) 的字符集标识; 它不适用于 MQDLH 结构本身中的字符数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。可以使用以下特殊值:

MQCCSI_INHERIT

此结构之后的数据中的字符数据与此结构位于同一字符集中。

队列管理器将消息中发送的结构中的此值更改为结构的实际字符集标识。如果未发生错误, 那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER, 那么不能使用 MQCCSI_INHERIT。

此值在以下环境中受支持: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

此字段的初始值为 MQCCSI_UNDEFINED。

DestQMgr 名称 (MQCHAR48)

DestQMgr 名称是作为消息原始目标的队列管理器的名称。

此字段的长度由 MQ_Q_MGR_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串, 在其他编程语言中为 48 个空白字符。

DestQName (MQCHAR48)

DestQName 是作为消息原始目标的消息队列的名称。

此字段的长度由 MQ_Q_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

Encoding (MQLONG)

编码是遵循 MQDLH 结构的数据 (通常是来自原始消息的数据) 的数字编码; 它不适用于 MQDLH 结构本身中的数字数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。

此字段的初始值为 0。

Format (MQCHAR8)

Format 是遵循 MQDLH 结构的数据的格式名称 (通常是来自原始消息的数据)。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。用于对此字段进行编码的规则与用于在 MQMD 中对 *Format* 字段进行编码的规则相同。

此字段的长度由 MQ_FORMAT_LENGTH 指定。此字段的初始值为 MQFMT_NONE。

PutAppl 名称 (MQCHAR28)

PutAppl 名称是将消息放入死信 (undelid-message) 队列的应用程序的名称。

名称的格式取决于 *PutApplType* 字段。格式可以因发行版而异。请参阅 [第 364 页的『MQMD - 消息描述符』](#) 中 *PutApplName* 字段的描述。

如果队列管理器将消息重定向到死信队列, 那么 *PutApplName* 包含队列管理器名称的前 28 个字符, 必要时填充空白。

此字段的长度由 MQ_PUT_APPL_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串, 在其他编程语言中为 28 个空白字符。

PutApplType (MQLONG)

PutAppl 类型是将消息放入死信 (undelivered-message) 队列的应用程序类型。

此字段与消息描述符 MQMD 中的 *PutApplType* 字段具有相同的含义 (请参阅 [第 364 页的『MQMD - 消息描述符』](#) 以获取详细信息)。

如果队列管理器将消息重定向到死信队列, 那么 *PutApplType* 的值为 MQAT_QMGR。

此字段的初始值为 0。

PutDate (MQCHAR8)

PutDate 是将消息放入死信 (undelivered-message) 队列的日期。

队列管理器生成此字段的日期所使用的格式为:

- YYYYMMDD

其中字符表示:

YYYY

年 (四位数字)

MM

年月 (01 至 12)

DD

月日 (01 到 31)

格林威治标准时间 (GMT) 用于 *PutDate* 和 *PutTime* 字段, 前提是系统时钟精确设置为 GMT。

此字段的长度由 MQ_PUT_DATE_LENGTH 给出。此字段的初始值是 C 中的空字符串, 在其他编程语言中为 8 个空白字符。

PutTime (MQCHAR8)

PutTime 是将消息放入死信 (undelivered-message) 队列的时间。

队列管理器生成此字段时使用的格式为:

- HHMMSSSTH

其中字符表示:

HH

小时 (00 到 23)

MM

分钟 (00 到 59)

SS

秒 (00 到 59; 请参阅注释)

T

十分之一秒 (0 到 9)

H

百分之一秒 (0 到 9)

注: 如果系统时钟同步到非常准确的时间标准, 那么在极少数情况下, 可能会在 *PutTime* 中返回 60 或 61 的秒数。当将闰秒插入全局时间标准时, 会发生此情况。

格林威治标准时间 (GMT) 用于 *PutDate* 和 *PutTime* 字段, 前提是系统时钟精确设置为 GMT。

此字段的长度由 MQ_PUT_TIME_LENGTH 指定。此字段的初始值是 C 中的空字符串, 在其他编程语言中为 8 个空白字符。

原因 (MQLONG)

"原因" 字段标识将消息放在死信队列而不是原始目标队列上的原因。

这标识了将消息放在死信队列而不是原始目标队列上的原因。它应该是 MQFB_* 或 MQRC_* 值 (例如 MQRC_Q_FULL) 之一。请参阅第 364 页的『MQMD - 消息描述符』中 *Feedback* 字段的描述, 以获取可能出现的公共 MQFB_* 值的详细信息。

如果该值在 MQFB_IMS_FIRST 到 MQFB_IMS_LAST 的范围内, 那么可以通过从 *Reason* 字段的值中减去 MQFB_IMS_ERROR 来确定实际 IMS 错误代码。

某些 MQFB_* 值仅出现在此字段中。它们与已传输到死信队列的存储库消息, 触发器消息或传输队列消息相关。这些字段为:

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')

处理触发器消息的应用程序无法启动触发器消息的 *ApplId* 字段中指定的应用程序 (请参阅第 522 页的『MQTM-触发器消息』)。

在 z/OS 上, CKTI CICS 事务是处理触发器消息的应用程序的示例。

MQFB_APPL_TYPE_ERROR (X'0000010B')

处理触发器消息的应用程序无法启动应用程序, 因为触发器消息的 *ApplType* 字段无效 (请参阅第 522 页的『MQTM-触发器消息』)。

在 z/OS 上, CKTI CICS 事务是处理触发器消息的应用程序的示例。

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')

消息位于 SYSTEM.CLUSTER.TRANSMIT.QUEUE 用于使用 MQOO_BIND_ON_OPEN 选项打开的集群队列, 但用于将消息传输到目标队列的远程集群接收方通道已删除, 然后才能发送消息。由于指定了 MQOO_BIND_ON_OPEN, 因此只能使用打开队列时选择的通道来传输消息。由于此通道不再可用, 因此消息将放置在死信队列上。

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')

该消息不是存储库消息。

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')

消息已由通道自动定义出口停止。

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')

消息已由通道消息出口停止。

MQFB_TM_ERROR (X'0000010A')

MQMD 中的 *Format* 字段指定 MQFMT_TRIGGER，但消息未以有效的 MQTM 结构开头。例如，*StrucId* 助记符可能无效，*Version* 可能无法识别，或者触发器消息的长度可能不足以包含 MQTM 结构。

在 z/OS 上，CKTI CICS 事务是处理触发器消息并可生成此反馈代码的应用程序的示例。

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')

消息通道代理程序发现传输队列上的消息的格式不正确。消息通道代理程序使用此反馈代码将消息放在死信队列上。

此字段的初始值为 MQRC_NONE。

StrucId (MQCHAR4)

StrucId 是结构标识。

该值必须为：

MQDLH_STRUC_ID

死信头结构的标识。

对于 C 编程语言，还定义了常量 MQDLH_STRUC_ID_ARRAY；此值与 MQDLH_STRUC_ID 相同，但是字符数组而不是字符串。

此字段的初始值为 MQDLH_STRUC_ID。

Version (MQLONG)

版本是结构版本号。

该值必须为：

MQDLH_VERSION_1

死信头结构的版本号。

以下常量指定当前版本的版本号：

MQDLH_CURRENT_VERSION

死信头结构的当前版本。

此字段的初始值为 MQDLH_VERSION_1。

MQDLH 的初始值和语言声明

| 表 496: MQDLH 的 MQDLH 中字段的初始值 | | |
|------------------------------|------------------|---------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQDLH_STRUC_ID | 'DLH~' |
| <i>Version</i> | MQDLH_VERSION_1 | 1 |
| <i>Reason</i> | MQRC_NONE | 0 |
| <i>DestQName</i> | None | 空字符串或空白 |
| <i>DestQMgrName</i> | None | 空字符串或空白 |
| <i>Encoding</i> | None | 0 |
| <i>CodedCharSetId</i> | MQCCSI_UNDEFINED | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>PutApplType</i> | None | 0 |
| <i>PutApplName</i> | None | 空字符串或空白 |

表 496: MQDLH 的 MQDLH 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|---------|-------|---------|
| PutDate | None | 空字符串或空白 |
| PutTime | None | 空字符串或空白 |

注意:

1. 符号 `␣` 表示单个空白字符。
2. 值 `Null` 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。
3. 在 C 编程语言中，宏变量 `MQDLH_DEFAULT` 包含上面列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQDLH MyDLH = {MQDLH_DEFAULT};
```

C 声明

```
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 声明

```
** 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).

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

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)

```

Visual Basic 声明

```
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-删除消息句柄选项

下表汇总了结构中的字段。

| 表 497: MQDMHO 中的字段 | | |
|--------------------|-------|-------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 选项 | 选项 |

MQDMHO 概述

可用性: 所有 WebSphere MQ 系统和 WebSphere MQ 客户机。

用途: MQDMHO 结构允许应用程序指定用于控制如何删除消息句柄的选项。该结构是 MQDLTMH 调用上的输入参数。

字符集和编码: MQDMHO 中的数据必须位于应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQDMHO 的字段

MQDMHO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

选项 (MQLONG)

该值必须为:

MQDMHO_NONE

未指定任何选项。

这始终是一个输入字段。此字段的初始值为 MQDMHO_NONE。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQDMHO_STRUC_ID

删除消息句柄选项结构的标识。

对于 C 编程语言，还定义了常量 `MQDMHO_STRUC_ID_ARRAY`；此值与 `MQDMHO_STRUC_ID` 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 `MQDMHO_STRUC_ID`。

Version (MQLONG)

这是结构版本号；值必须为：

MQDMHO_VERSION_1

Version-1 删除消息句柄选项结构。

以下常量指定当前版本的版本号：

MQDMHO_CURRENT_VERSION

当前版本的删除消息句柄选项结构。

这始终是一个输入字段。此字段的初始值为 `MQDMHO_VERSION_1`。

MQDMHO 的初始值和语言声明

| 表 498: MQDMHO 中字段的初始值 | | |
|-----------------------|-------------------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | <code>MQDMHO_STRUC_ID</code> | 'DMHO' |
| <i>Version</i> | <code>MQDMHO_VERSION_1</code> | 1 |
| <i>Options</i> | <code>MQDMHO_NONE</code> | 0 |

注意：

- 在 C 编程语言中，宏变量 `MQDMHO_DEFAULT` 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值：

```
MQDMHO MyDMHO = {MQDMHO_DEFAULT};
```

C 声明

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    Options;          /* Options that control the action of MQDLTMH */
};
```

COBOL 声明

```
** 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.
```

PL/I 声明

```
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 */

```

High Level Assembler 声明

```

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-删除消息属性选项

下表汇总了结构中的字段。MQDMPO 结构-删除消息属性选项

| 表 499: MQDMPO 中的字段 | | |
|--------------------|-----------------|-------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 控制 MQDMPO 操作的选项 | 选项 |

MQDMPO 概述

可用性: 所有 WebSphere MQ 系统和 WebSphere MQ 客户机。

用途:MQDMPO 结构允许应用程序指定用于控制如何删除消息属性的选项。该结构是 MQDLTMP 调用上的输入参数。

字符集和编码:MQDMPO 中的数据必须位于应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQDMPO 的字段

删除消息属性选项结构-字段

MQDMPO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

选项 (*MQLONG*)

删除消息属性选项结构-"选项" 字段

位置选项: 下列选项与属性相对于属性光标的相对位置相关。

MQDMPO_DEL_FIRST

删除与指定名称匹配的第一个属性。

MQDMPO_DEL_PROP_UNDER_CURSOR

删除属性游标指向的属性; 即上次使用 MQIMPO_INQ_FIRST 或 MQIMPO_INQ_NEXT 选项查询的属性。

复用消息句柄时, 将重置属性游标。当在 MQGET 调用上的 MQGMO 结构的 *MsgHandle* 字段或 MQPUT 调用上的 MQPMO 结构中指定消息句柄时, 也会重置此消息句柄。

如果在尚未建立属性游标时使用此选项, 那么调用将失败, 并返回完成代码 MQCC_FAILED 和原因 MQRC_PROPERTY_NOT_AVAILABLE。如果已删除属性游标所指向的属性, 那么调用也将失败, 并返回完成代码 MQCC_FAILED 和原因 MQRC_PROPERTY_NOT_AVAILABLE。

如果这两个选项都不需要, 那么可以使用以下选项:

MQDMPO_NONE

未指定任何选项。

此字段始终是输入字段。此字段的初始值为 MQDMPO_DEL_FIRST。

StrucId (MQCHAR4)

删除消息属性选项结构- StrucId 字段

这是结构标识。该值必须为:

MQDMPO_STRUC_ID

删除消息属性选项结构的标识。

对于 C 编程语言，还定义了常量 MQDMPO_STRUC_ID_ARRAY; 此值与 MQDMPO_STRUC_ID 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQDMPO_STRUC_ID。

Version (MQLONG)

删除消息属性选项结构-"版本" 字段

这是结构版本号。该值必须为:

MQDMPO_VERSION_1

删除消息属性选项结构的版本号。

以下常量指定当前版本的版本号:

MQDMPO_CURRENT_VERSION

当前版本的删除消息属性选项结构。

这始终是一个输入字段。此字段的初始值为 MQDMPO_VERSION_1。

MQDMPO 的初始值和语言声明

删除消息属性选项结构-初始值

| 表 500: MQDMPO 中字段的初始值 | | |
|-----------------------|--------------------|-------------|
| 字段名称 | 常量的名称 | 常量值 |
| StrucId | MQDMPO_STRUC_ID | 'DMPO' |
| Version | MQDMPO_VERSION_1 | 1 |
| Options | 用于控制 MQDLTMP 操作的选项 | MQDMPO_NONE |

注意:

- 在 C 编程语言中，宏变量 MQDMPO_DEFAULT 包含上面列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQDMPO MyDMPO = {MQDMPO_DEFAULT};
```

C 声明

删除消息属性选项结构-C 语言声明

```
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 声明

删除消息属性选项结构-COBOL 语言声明

```
** MQDMPO structure
   10 MQDMPO.
** Structure identifier
   15 MQDMPO-STRUCID          PIC X(4).
```

```

**      Structure version number
15 MQDPMO-VERSION          PIC S9(9) BINARY.
**      Options that control the action of MQDLTMP
15 MQDPMO-OPTIONS         PIC S9(9) BINARY.

```

PL/I 声明

删除消息属性选项结构-PL/I 语言声明

```

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 */

```

High Level Assembler 声明

删除消息属性选项结构-汇编程序语言声明

```

MQDPMO          DSECT
MQDPMO_STRUCID DS CL4 Structure identifier
MQDPMO_VERSION DS F  Structure version number
MQDPMO_OPTIONS DS F  Options that control the
*               action of MQDLTMP
MQDPMO_LENGTH  EQU *-MQDPMO
MQDPMO_AREA    DS CL(MQDPMO_LENGTH)

```

MQEPH-嵌入式 PCF 头

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|-----------------------|------------------------------|--------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>StrucLength</i> | MQEPH 结构及其之后的 MQCFH 和参数结构的长度 | StrucLength |
| <i>Encoding</i> | 上一个 PCF 参数结构之后数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | 上一个 PCF 参数结构之后数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | 上一个 PCF 参数结构之后数据的格式名称 | FORMAT |
| <i>Flags</i> | 标志 | 标志 |
| <i>PCFHeader</i> | 可编程命令格式 (PCF) 头 | PCFHeader |

MQEPH 概述

可用性: 所有 WebSphere MQ 平台。

用途:MQEPH 结构描述了当消息是可编程命令格式 (PCF) 消息时消息中存在的其他数据。 *PCFHeader* 字段定义遵循此结构的 PCF 参数, 这允许您将 PCF 消息数据与其他头一起遵循。

格式名:MQFMT_EMBEDDED_PCF

字符集和编码:MQEPH 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。

将 MQEPH 的字符集和编码设置到以下位置的 *CodedCharSetId* 和 *Encoding* 字段中:

- MQMD (如果 MQEPH 结构位于消息数据的开头), 或者
- MQEPH 结构之前的头结构 (所有其他情况)。

用法: 不能使用 MQEPH 结构将命令发送到命令服务器或任何其他队列管理器 PCF 接受服务器。

同样, 命令服务器或任何其他接受队列管理器 PCF 的服务器不会生成包含 MQEPH 结构的响应或事件。

MQEPH 的字段

MQEPH 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

CodedCharSetId (MQLONG)

这是遵循 MQEPH 结构和关联 PCF 参数的数据的字符集标识; 它不适用于 MQEPH 结构本身中的字符数据。

此字段的初始值为 MQCCSI_UNDEFINED。

Encoding (MQLONG)

这是遵循 MQEPH 结构和关联 PCF 参数的数据的数字编码; 它不适用于 MQEPH 结构本身中的字符数据。

此字段的初始值为 0。

Flags (MQLONG)

有下列值可用:

MQEPH_NONE

未指定任何标志。MQEPH_NONE 定义为帮助程序文档。不打算将此常量与任何其他常量一起使用, 但由于其值为零, 因此无法检测到此类使用。

MQEPH_CCSID_EMBEDDED

包含字符数据的参数的字符集在每个结构的 CodedCharSetId 字段中单独指定。StrucId 和 Format 字段的字符集由 MQEPH 结构之前的头结构中的 CodedCharSetId 字段定义, 或者由 MQMD 中的 CodedCharSetId 字段定义 (如果 MQEPH 位于消息的开头)。

此字段的初始值为 MQEPH_NONE。

Format (MQCHAR8)

这是遵循 MQEPH 结构和关联 PCF 参数的数据的格式名称。

此字段的初始值为 MQFMT_NONE。

PCFHeader (MQCFH)

这是可编程命令格式 (PCF) 头, 用于定义遵循 MQEPH 结构的 PCF 参数。这使您能够关注带有其他头的 PCF 消息数据。

最初使用以下值定义 PCF 头:

| 字段名称 | 常量的名称 | 常量值 |
|---------------------|--------------------|-----|
| <i>Type</i> | MQCFT_NONE | 0 |
| <i>StrucLength</i> | MQCFH_STRUC_LENGTH | 36 |
| <i>Version</i> | MQCFH_VERSION_3 | 3 |
| <i>StrucLength</i> | None | 0 |
| <i>Command</i> | MQCMD_NONE | 0 |
| <i>MsgSeqNumber</i> | None | 1 |
| <i>Control</i> | MQCFC_LAST | 1 |
| <i>CompCode</i> | MQCC_OK | 0 |
| <i>Reason</i> | MQRC_NONE | 0 |

| 字段名称 | 常量的名称 | 常量值 |
|-----------------------|-------|-----|
| <i>ParameterCount</i> | None | 0 |

应用程序必须将 Type 从 MQCFT_NONE 更改为有效的结构类型，以供其使用嵌入式 PCF 头。

StrucId (MQCHAR4)

该值必须为:

MQEPH_STRUC_ID

分发头结构的标识。

对于 C 编程语言，还定义了常量 MQEPH_STRUC_ID_ARRAY; 此值与 MQDH_STRUC_ID 相同，但是字符数组而不是字符串。

此字段的初始值为 MQEPH_STRUC_ID。

StrucLength (MQLONG)

这是下一个头结构之前的数据量。其中包括:

- MQEPH 头的长度
- 头后面的所有 PCF 参数的长度
- 这些参数后的任何空白填充

StrucLength 必须是 4 的倍数。

结构的固定长度部分由 MQEPH_STRUC_LENGTH_FIXED 定义。

此字段的初始值为 68。

Version (MQLONG)

该值必须为:

MQEPH_VERSION_1

嵌入式 PCF 头结构的版本号。

以下常量指定当前版本的版本号:

MQCFH_VERSION_3

嵌入式 PCF 头结构的当前版本。

此字段的初始值为 MQEPH_VERSION_1。

MQEPH 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|-----------------------|--------------------------|--------|
| <i>StrucId</i> | MQEPH_STRUC_ID | 'EPH↵' |
| <i>Version</i> | MQEPH_VERSION_1 | 1 |
| <i>StrucLength</i> | MQEPH_STRUC_LENGTH_FIXED | 68 |
| <i>Encoding</i> | None | 0 |
| <i>CodedCharSetId</i> | MQCCSI_UNDEFINED | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQEPH_NONE | 0 |
| <i>PCFHeader</i> | 第 320 页的表 502 中定义的名称和值 | 0 |

表 503: MQEPH 的 MQEPH 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|---|-------|-----|
| 注意: | | |
| 1. 符号 <code>\n</code> 表示单个空白字符。 2. 在 C 编程语言中，宏变量 <code>MQEPH_DEFAULT</code> 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值： | | |
| <pre>MQEPH MyEPH = {MQEPH_DEFAULT};</pre> | | |

C 声明

```
typedef struct tagMQEPH MQEPH;
struct tagMQEPH {
    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 声明

```
** 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-STRUCLNGTH 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-STRUCLNGTH 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
```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

MQEPH                DSECT
MQEPH_STRUCID        DS CL4  Structure identifier
MQEPH_VERSION        DS F    Structure version number
MQEPH_STRUCLNGTH     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_STRUCLNGTH 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)

```

Visual Basic 声明

```

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-Get-消息选项

下表汇总了结构中的字段。

| 表 504: MQGMO 中的字段 | | |
|---|-------------------------|--------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 用于控制 MQGET 的操作的选项 | MQGMO-"选项" 字段 |
| <i>WaitInterval</i> | 等待时间间隔 | WaitInterval |
| <i>Signal1</i> | 信号(F) | Signal1 |
| <i>Signal2</i> | 信号标识 | Signal2 |
| <i>ResolvedQName</i> | 目标队列的已解析名称 | ResolvedQName |
| 注: 如果 <i>Version</i> 小于 MQGMO_VERSION_2, 那么将忽略其余字段。 | | |
| <i>MatchOptions</i> | 控制用于 MQGET 的选择标准的选项 | MatchOptions |
| <i>GroupStatus</i> | 指示检索到的消息是否在组中的标志 | GroupStatus |
| <i>SegmentStatus</i> | 指示检索到的消息是否是逻辑消息段的标志 | SegmentStatus |
| <i>Segmentation</i> | 指示是否允许对检索到的消息进一步分段的标志 | 分段 |
| <i>Reserved1</i> | 已保留 | Reserved1 |
| 注: 如果 <i>Version</i> 小于 MQGMO_VERSION_3, 那么将忽略其余字段。 | | |
| <i>MsgToken</i> | 消息令牌 | MsgToken |
| <i>ReturnedLength</i> | 所返回消息数据的长度 (字节) | ReturnedLength |
| 注: 如果 <i>Version</i> 小于 MQGMO_VERSION_4, 那么将忽略其余字段。 | | |
| <i>Reserved2</i> | 已保留 | Reserved2 |
| <i>MsgHandle</i> | 消息句柄, 将在其中填充从队列检索的消息属性。 | MsgHandle |

MQGMO 概述

可用性: 所有 WebSphere MQ 平台。

用途: MQGMO 结构允许应用程序控制如何从队列中除去消息。此结构是 MQGET 调用上的输入/输出参数。

版本: MQGMO 的当前版本为 MQGMO_VERSION_4。某些字段仅在特定版本的 MQGMO 中可用。如果需要多个环境之间移植应用程序, 那么必须确保 MQGMO 的版本在所有环境之间一致。仅在结构的特定版本中存在的字段在 [第 324 页的『MQGMO-Get-消息选项』](#) 和字段描述中标识为此类字段。

为受支持的编程语言提供的头 COPY 和 INCLUDE 文件包含环境支持的最新版本的 MQGMO，但 *Version* 字段的初始值设置为 MQGMO_VERSION_1。要使用 version-1 结构中不存在的字段，请将 *Version* 字段设置为所需版本的版本号。

字符集和编码:MQGMO 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是，如果应用程序作为 MQ MQI 客户机运行，那么该结构必须采用客户机的字符集和编码。

MQGMO 的字段

MQGMO 结构包含以下字段; 这些字段以 **字母顺序**描述:

GroupStatus (MQCHAR)

此标志指示检索的消息是否在组中。

它具有下列其中一个值:

MQGS_NOT_IN_GROUP

消息不在组中。

MQGS_MSG_IN_GROUP

消息位于组中，但不是组中的最后一个消息。

MQGS_LAST_MSG_IN_GROUP

消息是组中的最后一个消息。

如果组仅包含一条消息，那么这也是返回的值。

这是输出字段。此字段的初始值为 MQGS_NOT_IN_GROUP。如果 *Version* 小于 MQGMO_VERSION_2，那么将忽略此字段。

MatchOptions (MQLONG)

这些选项允许应用程序选择 *MsgDesc* 参数中的哪些字段用于选择 MQGET 调用返回的消息。应用程序在此字段中设置必需选项，然后将 *MsgDesc* 参数中的相应字段设置为这些字段所需的值。只有在消息的 MQMD 中具有这些值的消息才是在 MQGET 调用上使用 *MsgDesc* 参数进行检索的候选者。选择要返回的消息时，将忽略未指定相应匹配选项的字段。如果在 MQGET 调用上未指定选择标准 (即，*any* 消息可接受)，请将 *MatchOptions* 设置为 MQMO_NONE。

- 在 z/OS 上，可以使用的选择标准可能受到用于队列的索引类型的限制。请参阅 *IndexType* 队列属性以获取更多详细信息。

如果指定 MQGMO_LOGICAL_ORDER，那么下一次 MQGET 调用仅可返回某些消息:

- 如果没有当前组或逻辑消息，那么只有 *MsgSeqNumber* 等于 1 且 *Offset* 等于 0 的消息才有资格返回。在这种情况下，您可以使用以下一个或多个匹配选项来选择返回哪些符合条件的消息:

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

- 如果存在当前组或逻辑消息，那么只有组中的下一条消息或逻辑消息中的下一个段才适合返回，并且不能通过指定 MQMO_* 选项来改变此情况。

在上述两种情况下，您都可以指定不适用的匹配选项，但 *MsgDesc* 参数中相关字段的值必须与要返回的消息中相应字段的值匹配; 调用失败，原因码为 MQRC_MATCH_OPTIONS_ERROR。

如果指定 MQGMO_MSG_UNDER_CURSOR 或 MQGMO_BROWSE_MSG_UNDER_CURSOR，那么将忽略 *MatchOptions*。

未使用匹配选项来获取基于消息属性的消息; 有关更多信息，请参阅第 422 页的『[SelectionString \(MQCHARV\)](#)』。

可以指定以下一个或多个匹配选项:

MQMO_MATCH_MSG_ID

要检索的消息必须具有与 MQGET 调用的 *MsgDesc* 参数中 *MsgId* 字段的值匹配的消息标识。此匹配是对可能应用的任何其他匹配项 (例如, 相关标识) 的补充。

如果省略此选项, 那么将忽略 *MsgDesc* 参数中的 *MsgId* 字段, 并且任何消息标识都将匹配。

注: 消息标识 MQMI_NONE 是与消息的 MQMD 中的任何消息标识匹配的特殊值。因此, 使用 MQMI_NONE 指定 MQMO_MATCH_MSG_ID 与不指定 MQMO_MATCH_MSG_ID 相同。

MQMO_MATCH_CORREL_ID

要检索的消息必须具有与 MQGET 调用的 *MsgDesc* 参数中 *CorrelId* 字段的值匹配的相关标识。此匹配是对可能应用的任何其他匹配项 (例如, 消息标识) 的补充。

如果省略此选项, 那么将忽略 *MsgDesc* 参数中的 *CorrelId* 字段, 并且任何相关标识都将匹配。

注: 相关标识 MQCI_NONE 是与消息的 MQMD 中的任何相关标识匹配的特殊值。因此, 使用 MQCI_NONE 指定 MQMO_MATCH_CORREL_ID 与不指定 MQMO_MATCH_CORREL_ID 相同。

MQMO_MATCH_GROUP_ID

要检索的消息必须具有与 MQGET 调用的 *MsgDesc* 参数中 *GroupId* 字段的值匹配的组标识。此匹配是对可能应用的任何其他匹配项 (例如, 相关标识) 的补充。

如果省略此选项, 那么将忽略 *MsgDesc* 参数中的 *GroupId* 字段, 并且任何组标识都将匹配。

注: 组标识 MQGI_NONE 是与消息的 MQMD 中的任何组标识匹配的特殊值。因此, 使用 MQGI_NONE 指定 MQMO_MATCH_GROUP_ID 与不指定 MQMO_MATCH_GROUP_ID 相同。

MQMO_MATCH_MSG_SEQ_NUMBER

要检索的消息必须具有与 MQGET 调用的 *MsgDesc* 参数中的 *MsgSeqNumber* 字段值匹配的消息序号。此匹配项是对可能应用的任何其他匹配项 (例如, 组标识) 的补充。

如果省略此选项, 那么将忽略 *MsgDesc* 参数中的 *MsgSeqNumber* 字段, 并且任何消息序号都将匹配。

MQMO_MATCH_OFFSET

要检索的消息必须具有与 MQGET 调用的 *MsgDesc* 参数中 *Offset* 字段的值匹配的偏移量。此匹配项是对可能应用的任何其他匹配项 (例如, 消息序号) 的补充。

如果未指定此选项, 那么将忽略 *MsgDesc* 参数中的 *Offset* 字段, 并且任何偏移量都将匹配。

- 此选项在 z/OS 上不受支持。

MQMO_MATCH_MSG_TOKEN

要检索的消息必须具有与 MQGET 调用上指定的 MQGMO 结构中 *MsgToken* 字段的值匹配的消息令牌。

可以对所有本地队列指定此选项。如果为 *IndexType* 为 MQIT_MSG_TOKEN (WLM 管理的队列) 的队列指定此参数, 那么不能使用 MQMO_MATCH_MSG_TOKEN 指定其他匹配选项。

不能将 MQMO_MATCH_MSG_TOKEN 与 MQGMO_WAIT 或 MQGMO_SET_SIGNAL 一起指定。如果应用程序要等待消息到达 *IndexType* 为 MQIT_MSG_TOKEN 的队列, 请指定 MQMO_NONE。

如果省略此选项, 那么将忽略 MQGMO 中的 *MsgToken* 字段, 并且任何消息令牌都将匹配。

如果未指定所描述的任何选项, 那么可以使用以下选项:

MQMO_NONE

在选择要返回的消息时不使用任何匹配项; 队列上的所有消息都适合检索 (但受 MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE 和 MQGMO_COMPLETE_MSG 选项的控制)。

MQMO_NONE 帮助程序文档。不打算将此选项与任何其他 MQMO_* 选项一起使用, 但由于其值为零, 因此无法检测到此类使用。

这是一个输入字段。此字段的初始值是具有 MQMO_MATCH_CORREL_ID 的 MQMO_MATCH_MSG_ID。如果 *Version* 小于 MQGMO_VERSION_2, 那么将忽略此字段。

注: 定义 *MatchOptions* 字段的初始值是为了与较早的 MQSeries 队列管理器兼容。但是, 在不使用选择标准从队列中读取一系列消息时, 此初始值要求应用程序在每次 MQGET 调用之前将 *MsgId* 和 *CorrelId*

字段重置为 MQMI_NONE 和 MQCI_NONE。避免需要通过将 *Version* 设置为 MQGMO_VERSION_2 并将 *MatchOptions* 设置为 MQMO_NONE 来重置 *MsgId* 和 *CorrelId*。

MsgHandle (MQHMSG)

如果指定了 MQGMO_PROPERTIES_AS_Q_DEF 选项，并且 *PropertyControl* 队列属性未设置为 MQPROP_FORCE_MQRFH2，那么这是将使用从队列中检索的消息的属性填充的消息的句柄。此句柄由 MQCRTMH 调用创建。在检索消息之前，将清除已与句柄关联的任何属性。

还可以指定以下值：

MQHM_NONE

未提供消息句柄。

如果提供了有效的消息句柄并在输出中使用该消息句柄来包含消息属性，那么 MQGET 调用上不需要消息描述符，与该消息句柄相关联的消息描述符用于输入字段。

如果在 MQGET 调用上指定了消息描述符，那么它始终优先于与消息句柄关联的消息描述符。

如果指定了 MQGMO_PROPERTIES_FORCE_MQRFH2，或者指定了 MQGMO_PROPERTIES_AS_Q_DEF，并且 *PropertyControl* 队列属性为 MQPROP_FORCE_MQRFH2，那么当未指定消息描述符参数时，调用将失败，原因码为 MQRC_MD_ERROR。

从 MQGET 调用返回时，将更新与此消息句柄相关联的属性和消息描述符，以反映所检索消息的状态（如果在 MQGET 调用上提供了消息描述符）。然后，可以使用 MQINQMP 调用来查询消息的属性。

除了消息描述符扩展外，如果存在可使用 MQINQMP 调用查询的属性，那么该属性不会包含在消息数据中；如果队列上的消息包含在消息数据中的属性，那么在将数据返回到应用程序之前，会将这些属性从消息数据中除去。

如果未提供消息句柄，或者版本低于 MQGMO_VERSION_4，那么必须在 MQGET 调用上提供有效的消息描述符。在受 MQGMO 结构和 *PropertyControl* 队列属性中的属性选项值限制的消息数据中返回任何消息属性（消息描述符中包含的属性除外）。

这始终是一个输入字段。此字段的初始值为 MQHM_NONE。如果 *Version* 小于 MQGMO_VERSION_4，那么将忽略此字段。

MsgToken (MQBYTE16)

MsgToken 字段-MQGMO 结构。此字段由队列管理器用于唯一地标识消息。

这是由队列管理器生成的字节字符串，用于在队列上唯一地标识消息。消息令牌在第一次将消息放在队列管理器上时生成，并与消息一起保留，直到从队列管理器中永久除去消息为止，除非重新启动队列管理器。

从队列中除去消息时，标识该消息实例的 *MsgToken* 不再有效，并且永不复用。如果重新启动队列管理器，那么重新启动前在队列上标识消息的 *MsgToken* 可能在重新启动后无效。但是，从不复用 *MsgToken* 来标识其他消息实例。*MsgToken* 由队列管理器生成，对于任何外部应用程序都不可见。

当通过调用 MQGET（其中提供了 V 3 或更高版本的 MQGMO）返回消息时，队列管理器将在 MQGMO 中返回标识队列上消息的 *MsgToken*。有一个例外：当从同步点外的队列中除去消息时，队列管理器可能不会返回 *MsgToken*，因为在后续 MQGET 调用上标识返回的消息无用处。应用程序仅应使用 *MsgToken* 来引用后续 MQGET 调用上的消息。

如果提供了 *MsgToken*，并且指定了 *MatchOption* MQMO_MATCH_MSG_TOKEN 并且未指定 MQGMO_MSG_UNDER_CURSOR 或 MQGMO_BROWSE_MSG_UNDER_CURSOR，那么只能返回由该 *MsgToken* 标识的消息。该选项在所有本地队列上都有效，而不考虑 INDXTYPE，并且在 z/OS 上，必须仅在工作负载管理器 (WLM) 队列上使用 INDXTYPE (MSGTOKEN)。

将检查指定的任何其他 *MatchOptions*，如果它们不匹配，那么将返回 MQRC_NO_MSG_AVAILABLE。If MQGMO_BROWSE_NEXT is coded with MQMO_MATCH_MSG_TOKEN, the message identified by the *MsgToken* is returned only if it is beyond the browse-cursor for the calling handle.

如果指定了 MQGMO_MSG_UNDER_CURSOR 或 MQGMO_BROWSE_MSG_UNDER_CURSOR，那么将忽略 MQMO_MATCH_MSG_TOKEN。

MQMO_MATCH_MSG_TOKEN 对于以下获取消息选项无效：

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

对于指定 MQMO_MATCH_MSG_TOKEN 的 MQGET 调用，必须向该调用提供 V 3 或更高版本的 MQGMO，否则将返回 MQRC_不法_gmo_version。

如果此时 *MsgToken* 无效，那么将返回带有 MQRC_NO_MSG_AVAILABLE 的 MQCC_FAILED，除非存在其他错误。

选项 (MQLONG)

MQGMO 选项控制 MQGET 的操作。可以指定零个或多个选项。如果需要多个可选值：

- 添加值 (请勿多次添加相同的常量)，或者
- 使用按位 OR 运算组合值 (如果编程语言支持位运算)。

将记录无效选项的组合；所有其他组合都有效。

等待选项：以下选项与等待消息到达队列相关：

MQGMO_WAIT

应用程序将等待合适的消息到达。应用程序等待的最大时间在 *WaitInterval* 中指定。

要点：如果有合适的消息立即可用，那么不会等待或延迟。

如果 MQGET 个请求被禁止，或者 MQGET 个请求在等待时被禁止，那么将取消等待。无论队列上是否存在适当的消息，调用都将完成 MQCC_FAILED 和原因码 MQRC_GET_INHIBITED。

可以将 MQGMO_WAIT 与 MQGMO_BROWSE_FIRST 或 MQGMO_BROWSE_NEXT 选项配合使用。

如果多个应用程序正在同一共享队列上等待，那么以下规则将选择在合适的消息到达时激活哪个应用程序：

| 表 505: 用于在共享队列上激活 MQGET 调用的规则。 | | |
|--------------------------------|---------------------------|--|
| 正在等待激活的 MQGET 个调用数 | | 结果 |
| 使用 BROWSE 选项 | 没有 BROWSE 选项 ¹ | |
| None | 一个或多个 | 将激活一个不带 BROWSE 选项的 MQGET 调用。 |
| 一个或多个 | None | 将激活所有带有 BROWSE 选项的 MQGET 调用。 |
| 一个或多个 | 一个或多个 | 将激活一个不带 BROWSE 选项的 MQGET 调用。使用 BROWSE 选项激活的 MQGET 调用数不可预测。 |

如果没有 BROWSE 选项的多个 MQGET 调用正在同一队列上等待，那么将仅激活一个调用。队列管理器尝试按以下顺序优先处理正在等待的调用：

1. 只能由特定消息 (例如，具有特定 *MsgId* 和/或 *CorrelId* 的消息) 满足的特定 get-wait 请求。
2. 可由任何消息满足的常规 get-wait 请求。

注：

- 在第一类中，没有为更具体的 get-wait 请求提供额外的优先级。例如，同时指定 *MsgId* 和 *CorrelId* 的请求。
- 在任一类别中，都无法预测选择了哪个应用程序。特别是，等待时间最长的应用程序不一定是所选的应用程序。
- 操作系统的路径长度和优先级调度注意事项可能意味着操作系统优先级低于预期的等待应用程序检索消息。
- 也可能发生以下情况：未处于等待状态的应用程序会优先检索消息，而不是其中的消息。

在 z/OS 上，以下要点适用：

¹ 指定 MQGMO_LOCK 选项的 MQGET 调用被视为非浏览调用。

- 如果您希望应用程序在等待消息到达时继续执行其他工作，请考虑改为使用信号选项 (MQGMO_SET_SIGNAL)。但是，信号选项是特定于环境的；您要在不同环境之间移植的应用程序不得使用它。
- 如果有多个 MQGET 调用在等待同一消息，同时使用等待选项和信号选项，那么将平等地考虑每个等待调用。将 MQGMO_SET_SIGNAL 与 MQGMO_WAIT 一起指定时发生错误。将此选项与未完成信号的队列句柄一起指定也是一个错误。
- 如果对 *IndexType* 为 MQIT_MSG_TOKEN 的队列指定 MQGMO_WAIT 或 MQGMO_SET_SIGNAL，那么不允许选择条件。这表示：
 - 如果您正在使用 version-1 MQGMO，请在 MQGET 调用 MQMI_NONE 和 MQCI_NONE 上指定的 MQMD 中设置 *MsgId* 和 *CorrelId* 字段。
 - 如果要使用 version-2 或更高版本 MQGMO，请将 *MatchOptions* 字段设置为 MQMO_NONE。
- 对于共享队列上的 MQGET 调用，并且该调用是浏览请求或组消息的破坏性获取，并且 *MsgId* 或 *CorrelId* 都不匹配，那么将每隔 200 毫秒重新发出一次 MQGET 调用，直到合适的消息到达队列或等待时间间隔到期为止。

此方法会导致意外的处理开销，并且在很少添加消息时不是有效的消息检索方法。要避免浏览案例的此开销，请在 MQGET 调用上指定 *MsgId* (如果未建立索引或由 *MsgId* 建立索引) 或 *CorrelId* (如果由 *CorrelId* 建立索引) 匹配。

如果与 MQGMO_BROWSE_MSG_UNDER_CURSOR 或 MQGMO_MSG_UNDER_CURSOR 一起指定，那么将忽略 MQGMO_WAIT；不会发生任何错误。

MQGMO_NO_WAIT

如果没有合适的消息可用，那么应用程序不会等待。MQGMO_NO_WAIT 与 MQGMO_WAIT 相反。定义 MQGMO_NO_WAIT 以帮助程序文档。如果两者都未指定，那么它是缺省值。

MQGMO_SET_SIGNAL

将此选项与 *Signal1* 和 *Signal2* 字段配合使用。它允许应用程序在等待消息到达时继续执行其他工作。它还允许 (如果合适的操作系统设施可用) 应用程序等待消息到达多个队列。

注：MQGMO_SET_SIGNAL 选项特定于环境；请勿将其用于要移植的应用程序。

在两种情况下，调用以与未指定此选项相同的方式完成：

1. 如果当前可用的消息满足消息描述符中指定的条件。
2. 如果检测到参数错误或其他同步错误。

如果当前没有满足消息描述符中指定的条件的消息可用，那么控制权将返回到应用程序，而不会等待消息到达。*CompCode* 和 *Reason* 参数设置为 MQCC_WARNING 和 MQRC_SIGNAL_REQUEST_ACCEPTED。未设置消息描述符中的其他输出字段以及 MQGET 调用的输出参数。当适当的信息稍后到达时，将通过发布 ECB 来传递该信号。

然后，调用者必须重新发出 MQGET 调用以检索消息。应用程序可以使用操作系统提供的功能来等待此信号。

如果操作系统提供了多重等待机制，那么您可以使用它来等待消息到达多个队列中的任何一个队列。

如果指定了非零 *WaitInterval*，那么将在等待时间间隔到期后传递信号。队列管理器还可以取消等待，在这种情况下，将传递信号。

多个 MQGET 调用可以为同一消息设置信号。应用程序的激活顺序与针对 MQGMO_WAIT 描述的顺序相同。

如果多个 MQGET 调用正在等待同一消息，那么将平等地考虑每个正在等待的调用。这些调用可以包含等待和信号选项的混合。

在某些情况下，MQGET 调用可以检索消息，并且可以传递来自同一消息到达的信号。在传递信号时，必须准备应用程序以确保没有消息可用。

队列句柄不能有多于一个未完成的信号请求。

此选项对于下列任何选项都无效：

- MQGMO_UNLOCK

- MQGMO_WAIT

对于共享队列上的 MQGET 调用，该调用是浏览请求或组消息的破坏性获取，并且 *MsgId* 或 *CorrelId* 都不匹配，用户的信号 ECB 将在 200 毫秒后发布 MQEC_MSG_ARRIVED。

即使在使用 MQEC_WAIT_INTERVAL_EXPIRED 发布队列时，在等待时间间隔到期之前，合适的消息可能尚未到达队列中，也会发生此情况。发布 MQEC_MSG_ARRIVED 时，必须重新发出另一个 MQGET 调用以检索消息 (如果有)。

此方法用于确保及时通知您消息到达，但在与非共享队列上的类似调用序列进行比较时，可能会显示为意外的处理开销。

当不经常添加消息时，这不是高效的检索方法。要避免浏览案例的此开销，请在 MQGET 调用上指定 *MsgId* (如果未建立索引或由 *MsgId* 建立索引) 或 *CorrelId* (如果由 *CorrelId* 建立索引) 匹配。

此选项仅在 z/OS 上受支持。

MQGMO_FAIL_IF QUIESCING

如果队列管理器处于停顿状态，那么强制 MQGET 调用失败。

在 z/OS 上，如果连接 (对于 CICS 或 IMS 应用程序) 处于停顿状态，那么此选项还会强制 MQGET 调用失败。

如果此选项与 MQGMO_WAIT 或 MQGMO_SET_SIGNAL 一起指定，并且在队列管理器进入停顿状态时等待或信号未完成：

- 已取消等待，调用将返回完成代码 MQCC_FAILED，原因码为 MQRC_Q_MGR QUIESCING 或 MQRC_CONNECTION QUIESCING。
- 使用特定于环境的信号完成代码取消信号。

在 z/OS 上，信号完成，事件完成代码为 MQEC_Q_MGR QUIESCING 或 MQEC_CONNECTION QUIESCING。

如果未指定 MQGMO_FAIL_IF QUIESCING，并且队列管理器或连接进入停顿状态，那么不会取消等待或信号。

同步点选项：以下选项与 MQGET 调用参与工作单元相关：

MQGMO_SYNCPOINT

请求是在正常工作单元协议中运行。该消息被标记为对其他应用程序不可用，但仅当落实工作单元时才会从队列中删除该消息。如果回退工作单元，那么该消息将再次可用。

您可以使 MQGMO_SYNCPOINT 和 MQGMO_NO_SYNCPOINT 保持未设置状态。在这种情况下，在工作单元协议中包含 get 请求由运行队列管理器的环境确定。它不是由运行应用程序的环境确定的。在 z/OS 上，获取请求位于工作单元中。在所有其他环境中，获取请求不在工作单元中。

由于这些差异，要移植的应用程序不得允许此选项为缺省值；请显式指定 MQGMO_SYNCPOINT 或 MQGMO_NO_SYNCPOINT。

此选项对于下列任何选项都无效：

- 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

该请求将在正常工作单元协议中运行，但仅前提是检索到的消息是持久的。持久消息在 MQMD 的 *Persistence* 字段中具有值 MQPER_PERSISTENT。

- 如果消息是持久消息，那么队列管理器会像应用程序已指定 MQGMO_SYNCPOINT 一样处理调用。

- 如果消息不是持久消息，那么队列管理器将处理调用，就像应用程序指定了 MQGMO_NO_SYNCPOINT 一样。

此选项对于下列任何选项都无效:

- 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

此选项在以下环境中受支持: AIX, HP-UX, z/OS, IBM i, Solaris 和 Linux, 以及连接到这些系统的 WebSphere MQ MQI 客户机。

MQGMO_NO_SYNCPOINT

请求是在正常工作单元协议之外运行。如果您在没有浏览选项的情况下收到消息，那么将立即从队列中删除该消息。无法通过回退工作单元使消息再次可用。

如果指定 MQGMO_BROWSE_FIRST 或 MQGMO_BROWSE_NEXT，那么将采用此选项。

您可以使 MQGMO_SYNCPOINT 和 MQGMO_NO_SYNCPOINT 保持未设置状态。在这种情况下，在工作单元协议中包含 get 请求由运行队列管理器的环境确定。它不是由运行应用程序的环境确定的。在 z/OS 上，获取请求位于工作单元中。在所有其他环境中，获取请求不在工作单元中。

由于这些差异，要移植的应用程序不得允许此选项为缺省值;请显式指定 MQGMO_SYNCPOINT 或 MQGMO_NO_SYNCPOINT。

此选项对于下列任何选项都无效:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

MQGMO_MARK_SKIP_BACKOUT

回退工作单元，而不在队列中恢复使用此选项标记的消息。

此选项仅在 z/OS 上受支持。

如果指定了此选项，那么还必须指定 MQGMO_SYNCPOINT。对于以下任何选项，MQGMO_MARK_SKIP_BACKOUT 都无效:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

注: 在 IMS 和 CICS 上，在回退包含标记为 MQGMO_MARK_SKIP_BACKOUT 的消息的工作单元之后，可能必须发出额外的 WebSphere MQ 调用。在落实包含标记的消息的新工作单元之前，必须发出 WebSphere MQ 调用。该调用可以是您喜欢的任何 WebSphere MQ 调用。

1. 在 IMS 上，如果尚未应用 IMS APAR PN60855，并且正在运行 IMS MPP 或 BMP 应用程序。
2. 在 CICS 上，如果您正在运行任何应用程序。

在这两种情况下，请在落实包含回退消息的新工作单元之前发出任何 WebSphere MQ 调用。

注: 在工作单元中, 只能有一个标记为跳过回退的获取请求, 以及没有或多个未标记的获取请求。

如果应用程序退出工作单元, 那么不会将使用 MQGMO_MARK_SKIP_BACKOUT 检索的消息复原到其先前状态。将回退其他资源更新。将该消息视为已在由回退请求启动的新工作单元中检索到该消息。将在不使用 MQGMO_MARK_SKIP_BACKOUT 选项的情况下检索消息。

如果在更改了某些资源之后, 工作单元显然无法成功完成, 那么 MQGMO_MARK_SKIP_BACKOUT 很有用。如果省略此选项, 那么回退工作单元将恢复队列上的消息。下次检索消息时, 将再次发生相同的事件序列。

但是, 如果在原始 MQGET 调用上指定 MQGMO_MARK_SKIP_BACKOUT, 那么回退工作单元会回退对其他资源的更新。将该消息视为已在新的工作单元下检索。应用程序可以执行相应的错误处理。它可以向原始消息的发送方发送报告消息, 或者将原始消息放在死信队列上。然后, 它可以落实新的工作单元。落实新的工作单元将从原始队列中永久除去消息。

MQGMO_MARK_SKIP_BACKOUT 标记单个物理消息。如果消息属于消息组, 那么不会标记该组中的其他消息。同样, 如果标记的消息是逻辑消息的段, 那么不会标记逻辑消息中的其他段。

可以标记组中的任何消息, 但如果使用 MQGMO_LOGICAL_ORDER 检索消息, 那么标记组中的第一条消息是有利的。如果回退工作单元, 那么会将第一条 (标记的) 消息移至新的工作单元。该组中的第二条和更高版本的消息将在队列中恢复。队列上剩余的消息无法由另一个使用 MQGMO_LOGICAL_ORDER 的应用程序检索。组中的第一条消息不再位于队列中。但是, 支持工作单元的应用程序可以使用 MQGMO_LOGICAL_ORDER 选项将第二条和更高版本的消息检索到新的工作单元中。已检索到第一条消息。

有时, 您可能需要回退新的工作单元。例如, 因为死信队列已满, 并且不得废弃消息。回退新的工作单元将恢复原始队列上的消息, 这将防止消息丢失。但是, 在此情况下, 无法继续处理。在回退新的工作单元后, 应用程序必须通知操作员或管理员存在不可恢复错误, 然后完成。

仅当包含 get 请求的工作单元被应用程序回退中断时, MQGMO_MARK_SKIP_BACKOUT 才有效。如果由于事务或系统失败而回退了包含获取请求的工作单元, 那么将忽略 MQGMO_MARK_SKIP_BACKOUT。使用此选项检索的任何消息都将以与不使用此选项检索的消息相同的方式在队列中恢复。

浏览选项: 以下选项与浏览队列中的消息相关:

MQGMO_BROWSE_FIRST

当使用 MQOO_BROWSE 选项打开队列时, 将建立浏览光标, 该光标在逻辑上位于队列上的第一条消息之前。然后, 可以使用指定 MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT 或 MQGMO_BROWSE_MSG_UNDER_CURSOR 选项的 MQGET 调用以非破坏性方式从队列中检索消息。浏览光标会标记队列上的消息中的位置, 下一个 MQGET 调用 MQGMO_BROWSE_NEXT 将从该位置搜索合适的消息。

对于以下任何选项, MQGMO_BROWSE_FIRST 都无效:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

如果未打开队列以进行浏览, 那么这也是错误。

带有 MQGMO_BROWSE_FIRST 的 MQGET 调用将忽略浏览光标的先前位置。检索队列上满足消息描述符中指定的条件的第一条消息。消息保留在队列上, 浏览光标位于此消息上。

在此调用之后, 浏览光标位于已返回的消息上。在发出下一个带有 MQGMO_BROWSE_NEXT 的 MQGET 调用之前, 可能会从队列中除去该消息。在这种情况下, 浏览光标将保留在消息占用的队列中的位置, 即使该位置现在为空。

将 MQGMO_MSG_UNDER_CURSOR 选项与非浏览 MQGET 调用配合使用, 以从队列中除去消息。

非浏览 MQGET 调用不会移动浏览光标，即使使用同一 *Hobj* 句柄也是如此。它也不会被返回完成代码 MQCC_FAILED 或原因码 MQRC_TRUNCATED_MSG_FAILED 的浏览 MQGET 调用移动。

指定带有此选项的 MQGMO_LOCK 选项，以锁定浏览的消息。

您可以使用 MQGMO_* 和 MQMO_* 选项的任何有效组合来指定 MQGMO_BROWSE_FIRST，这些选项用于控制对逻辑消息的组和段中的消息的处理。

如果指定 MQGMO_LOGICAL_ORDER，那么将按逻辑顺序浏览消息。如果省略该选项，那么将按物理顺序浏览消息。如果指定 MQGMO_BROWSE_FIRST，那么可以在逻辑顺序和物理顺序之间切换。使用 MQGMO_BROWSE_NEXT 的后续 MQGET 调用将与为队列句柄指定 MQGMO_BROWSE_FIRST 的最新调用相同的顺序浏览队列。

队列管理器保留 MQGET 调用的两组组和段信息。浏览调用的组和段信息与从队列中除去消息的调用的信息分别保留。如果指定 MQGMO_BROWSE_FIRST，那么队列管理器将忽略要浏览的组和段信息。它会像没有当前组和当前逻辑消息一样扫描队列。如果 MQGET 调用成功，完成代码为 MQCC_OK 或 MQCC_WARNING，那么用于浏览的组和段信息将设置为返回的消息的组和段信息。如果调用失败，那么组和段信息将与调用之前的组和段信息保持相同。

MQGMO_BROWSE_NEXT

将浏览光标前进到队列上满足 MQGET 调用上指定的选择标准的下一条消息。消息将返回到应用程序，但保留在队列中。

对于以下任何选项，MQGMO_BROWSE_NEXT 都无效：

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

如果未打开队列以进行浏览，那么这也是错误。

MQGMO_BROWSE_NEXT 的行为方式与 MQGMO_BROWSE_FIRST 相同，如果它是第一次调用以浏览队列，那么在打开队列进行浏览之后。

发出下一个带有 MQGMO_BROWSE_NEXT 的 MQGET 调用之前，可能会从队列中除去光标下的消息。浏览光标在逻辑上保留在消息占用的队列中的位置，即使该位置现在为空。

消息以两种方式之一存储在队列上：

- 优先级 (MQMDS_PRIORITY) 内的 FIFO，或
- FIFO 而不考虑 优先级 (MQMDS_FIFO)

MsgDeliverySequence 队列属性指示应用的方法 (请参阅 [第 731 页的『队列的属性』](#) 以获取详细信息)。

队列的 *MsgDeliverySequence* 可能为 MQMDS_PRIORITY。消息到达的队列的优先级高于浏览光标当前指向的优先级。在这种情况下，在使用 MQGMO_BROWSE_NEXT 的队列的当前清理期间找不到更高优先级的消息。只有在使用 MQGMO_BROWSE_FIRST 重置浏览光标后，或者通过重新打开队列时，才能找到该值。

如果需要，可以将 MQGMO_MSG_UNDER_CURSOR 选项与非浏览 MQGET 调用配合使用，以从队列中除去消息。

使用同一 *Hobj* 句柄的非浏览 MQGET 调用不会移动浏览光标。

使用此选项指定 MQGMO_LOCK 选项以锁定已浏览的消息。

您可以使用 MQGMO_* 和 MQMO_* 选项的任何有效组合来指定 MQGMO_BROWSE_NEXT，这些选项用于控制对逻辑消息的组和段中的消息的处理。

如果指定 `MQGMO_LOGICAL_ORDER`，那么将按逻辑顺序浏览消息。如果省略该选项，那么将按物理顺序浏览消息。如果指定 `MQGMO_BROWSE_FIRST`，那么可以在逻辑顺序和物理顺序之间切换。使用 `MQGMO_BROWSE_NEXT` 的后续 `MQGET` 调用将以与为队列句柄指定 `MQGMO_BROWSE_FIRST` 的最新调用相同的顺序浏览队列。如果不满足此条件，那么调用将失败，原因码为 `MQRC_INCONSISTENT_BROWSE`。

注：如果未指定 `MQGMO_LOGICAL_ORDER`，请在使用 `MQGET` 调用在消息组末尾以外进行浏览时特别小心。例如，假定组中的最后一条消息在队列中的组中的第一条消息之前。使用 `MQGMO_BROWSE_NEXT` 浏览到组末尾以外的位置，指定 `MQMO_MATCH_MSG_SEQ_NUMBER` 并将 `MsgSeqNumber` 设置为 1 将返回组中已浏览的第一条消息。此结果可能会立即发生，或者如果存在中间组，那么稍后会发生许多 `MQGET` 调用。同一注意事项适用于不在组中的逻辑消息。

浏览调用的组和段信息与从队列中除去消息的调用的信息分别保留。

MQGMO_BROWSE_MSG_UNDER_CURSOR

以非破坏性方式检索浏览光标指向的消息，而不考虑在 `MQGMO` 的 `MatchOptions` 字段中指定的 `MQMO_*` 选项。

对于以下任何选项，`MQGMO_BROWSE_MSG_UNDER_CURSOR` 都无效：

- `MQGMO_BROWSE_FIRST`
- `MQGMO_BROWSE_NEXT`
- `MQGMO_MARK_SKIP_BACKOUT`
- `MQGMO_MSG_UNDER_CURSOR`
- `MQGMO_SYNCPOINT`
- `MQGMO_SYNCPOINT_IF_PERSISTENT`
- `MQGMO_UNLOCK`

如果未打开队列以进行浏览，那么这也是错误。

浏览光标指向的消息是上次使用 `MQGMO_BROWSE_FIRST` 或 `MQGMO_BROWSE_NEXT` 选项检索的消息。如果自此队列打开以来未对此队列发出任何这些调用，那么该调用将失败。如果此后以破坏性方式检索了浏览光标下的消息，那么调用也将失败。

此调用不会更改浏览光标的位置。

`MQGMO_MSG_UNDER_CURSOR` 选项可以与非浏览 `MQGET` 调用配合使用，以从队列中除去消息。

非浏览 `MQGET` 调用不会移动浏览光标，即使使用同一 `Hobj` 句柄也是如此。它也不会被返回完成代码 `MQCC_FAILED` 或原因码 `MQRC_TRUNCATED_MSG_FAILED` 的浏览 `MQGET` 调用移动。

如果 `MQGMO_BROWSE_MSG_UNDER_CURSOR` 与 `MQGMO_LOCK` 一起指定：

- 如果已锁定消息，那么它必须是光标下的消息，因此返回该消息时不会再次解锁和锁定。消息仍处于锁定状态。
- 如果没有锁定消息，并且在浏览光标下有消息，那么会将其锁定并返回到应用程序。如果浏览光标下没有消息，那么调用将失败。

如果在不带 `MQGMO_LOCK` 的情况下指定 `MQGMO_BROWSE_MSG_UNDER_CURSOR`：

- 如果已锁定消息，那么它必须是光标下的消息。该消息将返回到应用程序，然后解锁。由于该消息现在已解锁，因此无法保证可以再次浏览该消息，或者由同一应用程序以破坏性方式检索该消息。另一个从队列获取消息的应用程序可能以破坏性方式检索了该消息。
- 如果没有锁定的消息，并且在浏览光标下有消息，那么会将其返回到应用程序。如果浏览光标下没有消息，那么调用将失败。

如果 `MQGMO_COMPLETE_MSG` 与 `MQGMO_BROWSE_MSG_UNDER_CURSOR` 一起指定，那么浏览光标必须标识 `MQMD` 中的 `Offset` 字段为零的消息。如果不满足此条件，那么调用将失败，原因码为 `MQRC_INVALID_MSG_UNDER_CURSOR`。

浏览调用的组和段信息与从队列中除去消息的调用的信息分别保留。

MQGMO_MSG_UNDER_CURSOR

检索浏览光标指向的消息，而不考虑 MQGMO 的 *MatchOptions* 字段中指定的 MQMO_* 选项。将从队列中除去该消息。

浏览光标指向的消息是上次使用 MQGMO_BROWSE_FIRST 或 MQGMO_BROWSE_NEXT 选项检索的消息。

如果 MQGMO_COMPLETE_MSG 与 MQGMO_MSG_UNDER_CURSOR 一起指定，那么浏览光标必须标识 MQMD 中的 *Offset* 字段为零的消息。如果不满足此条件，那么调用将失败，原因码为 MQRC_INVALID_MSG_UNDER_CURSOR。

此选项对于下列任何选项都无效：

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

如果未同时打开队列以进行浏览和输入，那么这也是错误。如果浏览光标当前未指向可检索的消息，那么 MQGET 调用将返回错误。

MQGMO_MARK_BROWSE_HANDLE

将标记由成功的 MQGET 返回的消息或由返回的 *MsgToken* 标识的消息。该标记特定于调用中使用的对象句柄。

未从队列中除去消息。

仅当还指定了下列其中一个选项时，MQGMO_MARK_BROWSE_HANDLE 才有效：

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

对于以下任何选项，MQGMO_MARK_BROWSE_HANDLE 都无效：

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

消息将保持此状态，直到发生下列其中一个事件为止：

- 相关的对象句柄已正常或以其他方式关闭。
- 通过使用选项 MQGMO_UNMARK_BROWSE_HANDLE 调用 MQGET 来对此句柄取消标记消息。
- 将从对破坏性 MQGET 的调用返回消息，该调用将使用 MQCC_OK 或 MQCC_WARNING 完成。即使稍后回滚 MQGET，消息状态也会保持更改。
- 消息将到期。

MQGMO_MARK_BROWSE_CO_OP

将针对协作集中的所有句柄标记由成功 MQGET 返回的消息或由返回的 *MsgToken* 标识的消息。

协作级别标记是对可能已设置的任何句柄级别标记的补充。

未从队列中除去消息。

仅当对指定了 MQOO_CO_OP 的 MQOPEN 的调用返回了所使用的对象句柄时，MQGMO_MARK_BROWSE_CO_OP 才有效。还必须指定下列其中一个 MQGMO 选项：

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

此选项对于下列任何选项都无效:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

如果已标记该消息, 并且未指定选项 MQGMO_UNMARKED_BROWSE_MSG, 那么调用将失败, 并返回 MQCC_FAILED 和原因码 MQRC_MSG_MARKED_BROWSE_CO_OP。

消息将保持此状态, 直到发生下列其中一个事件为止:

- 关闭协作集中的所有对象句柄。
- 通过使用选项 MQGMO_UNMARK_BROWSE_CO_OP 调用 MQGET 来取消对协作浏览器的标记消息。
- 此消息由队列管理器自动取消标记。
- 将从对非浏览 MQGET 的调用返回消息。即使稍后回滚 MQGET, 消息状态也会保持更改。
- 消息将到期。

MQGMO_UNMARKED_BROWSE_MSG

对 MQGET 的调用 (用于指定 MQGMO_UNMARKED_BROWSE_MSG) 将返回一条被视为未标记其句柄的消息。如果将消息标记为其句柄, 那么它不会返回消息。如果队列是通过调用 MQOPEN(使用选项 MQOO_CO_OP) 打开的, 并且消息已由协作集的成员标记, 那么它也不会返回消息。

此选项对于下列任何选项都无效:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP

调用指定此选项的 MQGET 后, 将不再考虑要为协作集标记的一组协作句柄中的任何打开句柄的消息。如果在此调用之前将消息标记为句柄级别, 那么该消息仍被视为标记为句柄级别。

仅当使用选项 MQOO_CO_OP 成功调用 MQOPEN 所返回的句柄时, 使用 MQGMO_UNMARK_BROWSE_CO_OP 才有效。即使消息未被视为由协作的句柄集标记, MQGET 也会成功。

MQGMO_UNMARK_BROWSE_CO_OP 在非浏览 MQGET 调用上无效, 或者对于以下任何选项无效:

- 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

在调用指定了此选项的 MQGET 之后, 不再将找到的消息视为被此句柄标记。

即使未对此句柄标记消息, 调用也会成功。

此选项对非浏览 MQGET 调用无效，或者对下列任何选项无效：

- 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

锁定选项：以下选项与队列上的锁定消息相关：

MQGMO_LOCK

锁定已浏览的消息，以使该消息对为队列打开的任何其他句柄不可见。仅当还指定了下列其中一个选项时，才能指定此选项：

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

对于每个队列句柄，只能锁定一条消息。消息可以是逻辑消息或物理消息：

- 如果指定 MQGMO_COMPLETE_MSG，那么构成逻辑消息的所有消息段都将锁定到队列句柄。这些消息必须全部存在于队列中，并且可供检索。
- 如果省略 MQGMO_COMPLETE_MSG，那么仅将单个物理消息锁定到队列句柄。如果此消息正好是逻辑消息的段，那么锁定段将阻止其他应用程序使用 MQGMO_COMPLETE_MSG 来检索或浏览逻辑消息。

锁定的消息始终是浏览光标下的消息。稍后指定 MQGMO_MSG_UNDER_CURSOR 选项的 MQGET 调用可将消息从队列中除去。使用队列句柄的其他 MQGET 调用也可以除去消息（例如，指定锁定消息的消息标识的调用）。

如果调用返回完成代码 MQCC_FAILED 或 MQCC_WARNING（原因码为 MQRC_TRUNCATED_MSG_FAILED），那么不会锁定任何消息。

如果应用程序未从队列中除去消息，那么将通过下列其中一项操作来释放锁定：

- 针对此句柄发出另一个 MQGET 调用，指定 MQGMO_BROWSE_FIRST 或 MQGMO_BROWSE_NEXT。如果调用通过 MQCC_OK 或 MQCC_WARNING 完成，那么将释放锁定。如果使用 MQCC_FAILED 完成调用，那么消息将保持锁定状态。但是，存在下列例外情况：
 - 如果随 MQRC_TRUNCATED_MSG_FAILED 一起返回了 MQCC_WARNING，那么不会解锁该消息。
 - 如果随 MQRC_NO_MSG_AVAILABLE 一起返回 MQCC_FAILED，那么将解锁该消息。

如果还指定了 MQGMO_LOCK，那么将锁定返回的消息。如果省略 MQGMO_LOCK，那么在调用后没有锁定消息。

如果指定 MQGMO_WAIT，并且没有立即可用的消息，那么将在启动等待之前解锁原始消息。

- 使用 MQGMO_BROWSE_MSG_UNDER_CURSOR 对此句柄发出另一个 MQGET 调用，而不使用 MQGMO_LOCK。如果调用通过 MQCC_OK 或 MQCC_WARNING 完成，那么将释放锁定。如果使用 MQCC_FAILED 完成调用，那么消息将保持锁定状态。但是，以下异常适用：
 - 如果随 MQRC_TRUNCATED_MSG_FAILED 一起返回了 MQCC_WARNING，那么不会解锁该消息。
- 使用 MQGMO_UNLOCK 对此句柄发出另一个 MQGET 调用。
- 使用句柄发出 MQCLOSE 调用。MQCLOSE 可能是隐式的，由应用程序结束导致。

指定 MQGMO_LOCK（MQOO_BROWSE 除外）不需要特殊的 MQOPEN 选项，而指定附带的浏览选项需要此选项。

对于以下任何选项，MQGMO_LOCK 都无效：

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

当您在 HP Integrity NonStop Server 上对 z/OS 队列管理器使用 IBM WebSphere MQ 客户机 (由 TMF 协调) 时, 无法使用 MQGMO_LOCK。

MQGMO_UNLOCK

要解锁的消息必须先前已由带有 MQGMO_LOCK 选项的 MQGET 调用锁定。如果没有针对此句柄锁定的消息, 那么调用将通过 MQCC_WARNING 和 MQRC_NO_MSG_LOCKED 完成。

如果指定 MQGMO_UNLOCK, 那么不会检查或变更 *MsgDesc*, *BufferLength*, *Buffer* 和 *DataLength* 参数。 *Buffer* 中未返回任何消息。

无需特殊打开选项即可指定 MQGMO_UNLOCK (尽管首先需要 MQOO_BROWSE 来发出锁定请求)。

此选项对除以下选项以外的任何选项都无效:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

无论是否指定了这两个选项, 都将采用这两个选项。

消息-数据选项: 以下选项与从队列中读取消息时消息数据的处理相关:

MQGMO_ACCEPT_TRUNCATED_MSG

如果消息缓冲区太小而无法保存完整消息, 请允许 MQGET 调用填充该缓冲区。 MQGET 将尽可能多的消息填充到缓冲区中。 它会发出警告完成代码, 并完成其处理。 这表示:

- 浏览消息时, 浏览光标将前进到返回的消息。
- 除去消息时, 将从队列中除去返回的消息。
- 如果未发生其他错误, 那么将返回原因码 MQRC_TRUNCATED_MSG_RECEIVED。

如果没有此选项, 那么缓冲区仍会填充尽可能多的消息。 发出了警告完成代码, 但未完成处理。 这表示:

- 浏览消息时, 浏览光标不高级。
- 除去消息时, 不会从队列中除去消息。
- 如果未发生其他错误, 那么将返回原因码 MQRC_TRUNCATED_MSG_FAILED。

MQGMO_CONVERT

此选项会将消息中的应用程序数据转换为符合 MQGET 调用的 *MsgDesc* 参数中指定的 *CodedCharSetId* 和 *Encoding* 值。 在将数据复制到 *Buffer* 参数之前, 将对其进行转换。

转换过程假定在放入消息时指定的 *Format* 字段用于标识消息中数据的性质。 消息数据由队列管理器针对内置格式进行转换, 并由用户编写的出口针对其他格式进行转换。 有关数据转换出口的详细信息, 请参阅 [第 791 页的『数据转换出口』](#)。

- 如果转换成功, 那么在从 MQGET 调用返回时, *MsgDesc* 参数中指定的 *CodedCharSetId* 和 *Encoding* 字段保持不变。
- 如果仅转换失败, 那么将返回未转换的消息数据。 *MsgDesc* 中的 *CodedCharSetId* 和 *Encoding* 字段将设置为未转换的消息的值。 在此情况下, 完成代码为 MQCC_WARNING。

在任一情况下, 这些字段都描述 *Buffer* 参数中返回的消息数据的字符集标识和编码。

请参阅 [第 364 页的『MQMD - 消息描述符』](#) 中描述的 *Format* 字段, 以获取队列管理器为其执行转换的格式名称的列表。

组和段选项: 以下选项与逻辑消息的组和段中的消息处理相关。 在选项描述之前, 以下是一些重要术语的定义:

物理消息

物理消息是可以放在队列上或从队列中除去的最小信息单元。它通常对应于在单个 MQPUT, MQPUT1 或 MQGET 调用上指定或检索的信息。每条物理消息都有自己的消息描述符 MQMD。通常,物理消息通过消息标识的不同值 (MQMD 中的 *MsgId* 字段) 进行区分。队列管理器不会强制实施不同的值。

逻辑消息

逻辑消息是应用程序信息的单个单元。在没有系统约束的情况下,逻辑消息与物理消息相同。如果逻辑消息很大,那么系统约束可能建议或需要将逻辑消息拆分为两个或多个物理消息(称为段)。

已分段的逻辑消息由两个或更多具有相同非空组标识的物理消息 (MQMD 中的 *GroupId* 字段) 组成。它们具有相同的消息序号,即 MQMD 中的 *MsgSeqNumber* 字段。通过 MQMD 中的段偏移量 *Offset* 字段的不同值来区分段。段偏移是物理消息中的数据从逻辑消息中的数据开始的偏移量。由于每个段都是物理消息,因此逻辑消息中的段通常具有不同的消息标识。

未分段但发送应用程序允许分段的逻辑消息也具有非空组标识。在这种情况下,如果逻辑消息不属于消息组,那么只有一条具有该组标识的物理消息。发送应用程序已禁止分段的逻辑消息具有空组标识 MQGI_NONE,除非逻辑消息属于消息组。

消息组

消息组是一组具有相同非空组标识的一个或多个逻辑消息。该组中的逻辑消息通过消息序号的不同值进行区分。序号是 1 到 n 范围内的整数,其中 n 是组中的逻辑消息数。如果对一条或多条逻辑消息进行分段,那么组中的物理消息数超过 n 条。

MQGMO_LOGICAL_ORDER

MQGMO_LOGICAL_ORDER 控制队列句柄的连续 MQGET 调用返回消息的顺序。必须在每个调用上指定该选项。

如果为同一队列句柄的连续 MQGET 调用指定了 MQGMO_LOGICAL_ORDER,那么将按消息序号的顺序返回组中的消息。逻辑消息的段按其段偏移量给定的顺序返回。此顺序可能与这些消息和段在队列中出现的顺序不同。

注: 指定 MQGMO_LOGICAL_ORDER 不会对不属于组且不属于段的消息产生负面影响。实际上,这类消息被视为属于仅由一条消息组成的消息组。从包含组,消息段和非组中的未分段消息的混合消息的队列中检索消息时,可以安全地指定 MQGMO_LOGICAL_ORDER。

要按所需顺序返回消息,队列管理器会在连续的 MQGET 调用之间保留组和段信息。组和段信息标识队列句柄的当前消息组和当前逻辑消息。它还标识组和逻辑消息中的当前位置,以及是否在工作单元中检索消息。由于队列管理器保留此信息,因此应用程序无需在每次 MQGET 调用之前设置组和段信息。具体而言,这意味着应用程序不需要在 MQMD 中设置 *GroupId*, *MsgSeqNumber* 和 *Offset* 字段。但是,应用程序必须在每个调用上正确设置 MQGMO_SYNCPOINT 或 MQGMO_NO_SYNCPOINT 选项。

当队列打开时,没有当前消息组和当前逻辑消息。当 MQGET 调用返回具有 MQMF_MSG_IN_GROUP 标志的消息时,消息组将成为当前消息组。在连续调用上指定了 MQGMO_LOGICAL_ORDER 时,该组将保留当前组,直到返回具有以下内容的消息为止:

- MQMF_LAST_MSG_IN_GROUP without MQMF_SEGMENT (即,组中的最后一条逻辑消息未分段),或者
- 带有 MQMF_LAST_SEGMENT 的 MQMF_LAST_MSG_IN_GROUP (即,返回的消息是组中最后一条逻辑消息的最后一段)。

返回此类消息时,将终止消息组,并且在成功完成 MQGET 调用时,不再存在当前组。以类似方式,当 MQGET 调用返回具有 MQMF_SEGMENT 标志的消息时,逻辑消息将成为当前逻辑消息。当返回具有 MQMF_LAST_SEGMENT 标志的消息时,将终止逻辑消息。

如果未指定选择标准,那么后续 MQGET 调用将以正确的顺序返回队列中第一个消息组的消息。然后,它们将返回第二个消息组的消息,依此类推,直到没有更多消息可用为止。可以通过在 *MatchOptions* 字段中指定以下一个或多个选项来选择返回的特定消息组:

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

但是,仅当没有当前消息组或逻辑消息时,这些选项才有效。请参阅第 324 页的『MQGMO-Get-消息选项』中描述的 *MatchOptions* 字段以获取更多详细信息。

第 340 页的表 506 显示了队列管理器在尝试查找要在 MQGET 调用上返回的消息时查找的 *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber* 和 *Offset* 字段的值。这些规则既适用于从队列中除去消息, 也适用于在队列中浏览消息。在表中, 表示 "是" 或 "否":

LOG ORD

指示是否在调用上指定了 MQGMO_LOGICAL_ORDER 选项。

Cur grp

指示在调用之前是否存在当前消息组。

Cur log msg

指示在调用之前是否存在当前逻辑消息。

其他列

显示队列管理器要查找的值。"先前" 表示针对队列句柄的先前消息中的字段返回的值。

| 指定的选项 | 调用前的组和 log-msg 状态 | | 队列管理器查找的值 | | | | |
|-------|-------------------|-------------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| | Cur grp | Cur log msg | MsgId | CorrelId | GroupId | MsgSeqNumber | Offset |
| Yes | 否 | 否 | 控制者 <i>MatchOptions</i> | 控制者 <i>MatchOptions</i> | 控制者 <i>MatchOptions</i> | 1 | 0 |
| Yes | 否 | Yes | 任何消息标识 | 任何相关标识 | 上一个组标识 | 1 | 前一个偏移量 + 前一个段长 |
| Yes | Yes | 否 | 任何消息标识 | 任何相关标识 | 上一个组标识 | 前一个序列号 + 1 | 0 |
| Yes | Yes | Yes | 任何消息标识 | 任何相关标识 | 上一个组标识 | 前一个序列号 | 前一个偏移量 + 前一个段长 |
| 否 | 任一 | 任一 | 控制者 <i>MatchOptions</i> | 控制者 <i>MatchOptions</i> | 控制者 <i>MatchOptions</i> | 控制者 <i>MatchOptions</i> | 控制者 <i>MatchOptions</i> |

如果队列上存在多个符合返回条件的消息组, 那么将按每个组中第一条逻辑消息的第一段在队列上的位置确定的顺序返回这些组。即, 消息序号为 1 且偏移量为 0 的物理消息确定返回合格组的顺序。

MQGMO_LOGICAL_ORDER 选项影响工作单元, 如下所示:

- 如果在工作单元中检索组中的第一个逻辑消息或段, 那么必须在工作单元中检索组中的所有其他逻辑消息和段(如果使用相同的队列句柄)。但是, 不需要在同一工作单元中检索这些信息。这允许在队列句柄的两个或多个连续工作单元之间拆分由许多物理消息组成的消息组。
- 如果未在工作单元中检索组中的第一个逻辑消息或段, 并且使用相同的队列句柄, 那么不能在工作单元中检索组中的任何其他逻辑消息和段。

如果不满足这些条件, 那么 MQGET 调用将失败, 原因码为 MQRC_INCONSISTENT_UOW。

指定 MQGMO_LOGICAL_ORDER 时, 在 MQGET 调用上提供的 MQGMO 不得小于 MQGMO_VERSION_2, 并且 MQMD 不得小于 MQMD_VERSION_2。如果未满足此条件, 那么调用将失败, 原因码为 MQRC_WRONG_GMO_VERSION 或 MQRC_WRONG_MD_VERSION(视情况而定)。

如果对队列句柄的连续 MQGET 调用未指定 MQGMO_LOGICAL_ORDER, 那么将返回消息而不考虑它们是属于消息组还是属于逻辑消息段。这意味着可能会返回来自特定组或逻辑消息的消息或段, 或者与来自其他组或逻辑消息的消息或段混合, 或者与不在组中且不是段的消息混合。在此情况下, 后续 MQGET 调用返回的特定消息由这些调用上指定的 MQMO_* 选项控制(请参阅第 324 页的『MQGMO-Get-消息选项』中描述的 *MatchOptions* 字段以获取这些选项的详细信息)。

这是在发生系统故障后，可用于在中间重新启动消息组或逻辑消息的方法。当系统重新启动时，应用程序可以将 *GroupId*, *MsgSeqNumber*, *Offset* 和 *MatchOptions* 字段设置为相应的值，然后在设置了 MQGMO_SYNCPOINT 或 MQGMO_NO_SYNCPOINT 的情况下发出 MQGET 调用，但未指定 MQGMO_LOGICAL_ORDER。如果此调用成功，那么队列管理器将保留组和段信息，并且使用该队列句柄的后续 MQGET 调用可正常指定 MQGMO_LOGICAL_ORDER。

队列管理器为 MQGET 调用保留的组和段信息与为 MQPUT 调用保留的组和段信息不同。此外，队列管理器还会保留以下项的单独信息：

- 从队列中除去消息的 MQGET 调用。
- MQGET 调用，用于浏览队列上的消息。

对于任何给定的队列句柄，应用程序可以将指定 MQGMO_LOGICAL_ORDER 的 MQGET 调用与不指定的 MQGET 调用混合使用。但是，请注意以下几点：

- 如果省略 MQGMO_LOGICAL_ORDER，那么每次成功的 MQGET 调用都会导致队列管理器将保存的组和段信息设置为与返回的消息对应的值；这将替换队列管理器为队列句柄保留的现有组和段信息。仅修改与调用操作（浏览或除去）相应的信息。
- 如果省略 MQGMO_LOGICAL_ORDER，那么如果存在当前消息组或逻辑消息，那么调用不会失败；调用可能成功，并带有 MQCC_WARNING 完成代码。第 341 页的表 507 显示可能出现的各种情况。在这些情况下，如果完成代码不是 MQCC_OK，那么原因码为下列其中一项（视情况而定）：
 - MQRC_INCOMPLETE_GROUP
 - MQRC_INCOMPLETE_MSG
 - MQRC_INCONSISTENT_UOW

注：队列管理器在浏览队列时，或在关闭已打开以进行浏览但未输入的队列时，不会检查组和段信息；在这些情况下，完成代码始终为 MQCC_OK（假定没有其他错误）。

| 当前调用是 | 先前的调用是 MQGET with MQGMO_LOGICAL_ORDER | 先前调用为 MQGET，但没有 MQGMO_LOGICAL_ORDER |
|--------------------------------|---------------------------------------|-------------------------------------|
| 带 MQGMO_LOGICAL_ORDER 的 MQGET | MQCC_FAILED | MQCC_FAILED |
| 不带 MQGMO_LOGICAL_ORDER 的 MQGET | MQCC_WARNING | MQCC_OK |
| MQCLOSE，使用未结束的组或逻辑消息 | MQCC_WARNING | MQCC_OK |

建议要按逻辑顺序检索消息和段的应用程序指定 MQGMO_LOGICAL_ORDER，因为这是要使用的最简单选项。该选项可使应用程序无需管理组和段信息，因为队列管理器会管理此信息。但是，与 MQGMO_LOGICAL_ORDER 选项提供的控制相比，专用应用程序可能需要更多的控制，而这可以通过不指定该选项来实现。然后，应用程序必须确保在每个 MQGET 调用之前正确设置 MQMD 中的 *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber* 和 *Offset* 字段以及 MQGMO 中的 *MatchOptions* 中的 MQMO_* 选项。

例如，要转发它接收的物理消息（而不考虑这些消息是在逻辑消息的组还是段中）的应用程序不得指定 MQGMO_LOGICAL_ORDER。在发送和接收队列管理器间具有多条路径的复杂网络中，物理消息到达时可能杂乱无序。通过在 MQPUT 调用上既不指定 MQGMO_LOGICAL_ORDER，也不指定相应的 MQPMO_LOGICAL_ORDER，转发应用程序可以在到达时立即检索和转发每条物理消息，而不必按逻辑顺序等待下一条消息到达。

您可以将 MQGMO_LOGICAL_ORDER 与任何其他 MQGMO_* 选项一起指定，并在适当情况下将其与各种 MQMO_* 选项一起指定（请参阅上文）。

- 在 z/OS 上，专用队列和共享队列支持此选项，但队列必须具有索引类型 MQIT_GROUP_ID。对于共享队列，队列映射到的 CFSTRUCT 对象必须处于 CFLEVEL (\$TAG1) 或 CFLEVEL (4)。

- 在 AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机上, 所有本地队列都支持此选项。

MQGMO_COMPLETE_MSG

MQGET 调用只能返回完整的逻辑消息。如果逻辑消息已分段, 那么队列管理器将重新组合这些段并将完整的逻辑消息返回给应用程序; 对于检索该逻辑消息的应用程序而言, 逻辑消息已分段这一事实并不明显。

注: 这是导致队列管理器重新组合消息段的唯一选项。如果未指定, 那么段将单独返回到应用程序 (如果它们存在于队列中) (并且它们满足在 MQGET 调用上指定的其他选择标准)。不希望接收个别段的应用程序必须始终指定 MQGMO_COMPLETE_MSG。

要使用此选项, 应用程序必须提供足以容纳完整消息的缓冲区, 或者指定 MQGMO_ACCEPT_TRUNCATED_MSG 选项。

如果队列包含缺少部分段的分段消息 (可能是因为它们在网络中已延迟并且尚未到达), 那么指定 MQGMO_COMPLETE_MSG 将阻止检索属于不完整逻辑消息的段。但是, 这些消息段仍构成 *CurrentQDepth* 队列属性的值; 这意味着可能没有可检索的逻辑消息, 即使 *CurrentQDepth* 大于零也是如此。

对于持久消息, 队列管理器只能在工作单元中重新组合段:

- 如果 MQGET 调用在用户定义的工作单元中运行, 那么将使用该工作单元。如果在重新组装过程中调用失败, 那么队列管理器会在队列中恢复在重新组装期间除去的任何段。但是, 此故障不会阻止成功落实工作单元。
- 如果调用在用户定义的工作单元外部运行, 并且不存在用户定义的工作单元, 那么队列管理器将在调用期间创建工作单元。如果调用成功, 那么队列管理器将自动落实工作单元 (应用程序不需要执行此操作)。如果调用失败, 那么队列管理器将回退工作单元。
- 如果调用在用户定义的工作单元外部运行, 但存在用户定义的工作单元, 那么队列管理器无法重新组合。如果消息不需要重新组装, 那么调用仍可成功。但是, 如果消息需要重新组装, 那么调用将失败, 原因码为 MQRC_UOW_NOT_AVAILABLE。

对于非持久消息, 队列管理器不需要工作单元即可执行重新组装。

作为段的每条物理消息都有自己的消息描述符。对于构成单个逻辑消息的段, 消息描述符中的大多数字段对于逻辑消息中的所有段都是相同的; 通常只有 *MsgId*, *Offset* 和 *MsgFlags* 字段在逻辑消息中的段之间有所不同。但是, 如果将段放在中间队列管理器中的死信队列上, 那么 DLQ 处理程序将检索指定 MQGMO_CONVERT 选项的消息, 这可能会导致更改段的字符集或编码。如果 DLQ 处理程序在途中成功发送段, 那么段可能具有与逻辑消息中的其他段不同的字符集或编码 (当段到达目标队列管理器时)。

由 *CodedCharSetId* 和 *Encoding* 字段不同的段组成的逻辑消息无法由队列管理器重新组合为单个逻辑消息。相反, 队列管理器会在逻辑消息开头重新组合并返回前几个连续段, 这些段具有相同的字符集标识和编码, 并且 MQGET 调用将完成代码为 MQCC_WARNING, 原因码为 MQRC_INCONSISTENT_CCSDS 或 MQRC_INCONSISTENT_ENCODINGS (视情况而定)。无论是否指定了 MQGMO_CONVERT, 都会发生此情况。要检索其余段, 应用程序必须在不使用 MQGMO_COMPLETE_MSG 选项的情况下重新发出 MQGET 调用, 逐个检索段。MQGMO_LOGICAL_ORDER 可用于按顺序检索其余段。

放置段的应用程序还可以将消息描述符中的其他字段设置为不同段之间不同的值。但是, 如果接收应用程序使用 MQGMO_COMPLETE_MSG 来检索逻辑消息, 那么执行此操作没有任何优势。当队列管理器重新组装逻辑消息时, 它会在消息描述符中返回来自第一个段的消息描述符的值; 唯一的例外是 *MsgFlags* 字段, 队列管理器设置此字段以指示重新组装的消息是唯一的段。

如果为报告消息指定了 MQGMO_COMPLETE_MSG, 那么队列管理器将执行特殊处理。队列管理器检查队列以查看与逻辑消息中的不同段相关的该报告类型的所有报告消息是否都存在于队列中。如果是, 那么可以通过指定 MQGMO_COMPLETE_MSG 将它们作为单一消息进行检索。要实现此目的, 必须由支持分段的队列管理器或 MCA 生成报告消息, 或者发端应用程序必须请求至少 100 字节的消息数据 (即, 必须指定相应的 MQRO_*_WITH_DATA 或 MQRO_*_WITH_FULL_DATA 选项)。如果存在的应用程序数据量小于段的完整数据量, 那么将在返回的报告消息中使用空值替换缺少的字节。

如果 MQGMO_COMPLETE_MSG 与 MQGMO_MSG_UNDER_CURSOR 或 MQGMO_BROWSE_MSG_UNDER_CURSOR 一起指定, 那么浏览光标必须位于 MQMD 中 *Offset* 字段的值为

0 的消息上。如果不满足此条件，那么调用将失败，原因码为 MQRC_INVALID_MSG_UNDER_CURSOR。

MQGMO_COMPLETE_MSG 意味着 MQGMO_ALL_SEGMENTS_AVAILABLE，因此不需要指定。

可以使用除 MQGMO_SYNCPOINT_IF_PERSISTENT 以外的任何其他 MQGMO_* 选项以及除 MQGMO_MATCH_OFFSET 以外的任何 MQMO_* 选项来指定 MQGMO_COMPLETE_MSG。

- 在 z/OS 上，专用队列和共享队列支持此选项，但队列的索引类型必须为 MQIT_GROUP_ID。对于共享队列，队列映射到的 CFSTRUCT 对象必须处于 CFLEVEL (\$TAG1) 或 CFLEVEL (4)。
- 在 AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机上，所有本地队列都支持此选项。

MQGMO_ALL_MSGS_AVAILABLE

仅当组中的所有消息都可用时，组中的消息才可供检索。如果队列包含缺少某些消息的消息组 (可能是因为这些消息在网络中已延迟并且尚未到达)，那么指定 MQGMO_ALL_MSGS_AVAILABLE 将阻止检索属于不完整组的消息。但是，这些消息仍构成 *CurrentQDepth* 队列属性的值；这意味着可能没有可检索的消息组，即使 *CurrentQDepth* 大于零也是如此。如果没有其他可检索的消息，那么将在指定的等待时间间隔 (如果有) 到期后返回原因码 MQRC_NO_MSG_AVAILABLE。

MQGMO_ALL_MSGS_AVAILABLE 的处理取决于是否还指定了 MQGMO_LOGICAL_ORDER：

- 如果同时指定了这两个选项，那么当没有当前组或逻辑消息时，MQGMO_ALL_MSGS_AVAILABLE 仅有效。如果存在当前组或逻辑消息，那么将忽略 MQGMO_ALL_MSGS_AVAILABLE。这意味着在按逻辑顺序处理消息时，MQGMO_ALL_MSGS_AVAILABLE 可以保持开启状态。
- 如果在不带 MQGMO_LOGICAL_ORDER 的情况下指定 MQGMO_ALL_MSGS_AVAILABLE，那么 MQGMO_ALL_MSGS_AVAILABLE 始终具有作用。这意味着必须在从队列中除去组中的第一条消息之后关闭该选项，以便能够除去组中的其余消息。

成功完成指定 MQGMO_ALL_MSGS_AVAILABLE 的 MQGET 调用意味着在发出 MQGET 调用时，组中的所有消息都在队列中。但是，请注意，其他应用程序仍可以从组中除去消息 (该组未锁定到检索组中第一条消息的应用程序)。

如果省略此选项，那么即使组不完整，也可以检索属于组的消息。

MQGMO_ALL_MSGS_AVAILABLE 意味着 MQGMO_ALL_SEGMENTS_AVAILABLE，因此不需要指定。

可以使用任何其他 MQGMO_* 选项以及任何 MQMO_* 选项来指定 MQGMO_ALL_MSGS_AVAILABLE。

- 在 z/OS 上，专用队列和共享队列支持此选项，但队列的索引类型必须为 MQIT_GROUP_ID。对于共享队列，队列映射到的 CFSTRUCT 对象必须处于 CFLEVEL (\$TAG1) 或 CFLEVEL (4)。
- 在 AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机上，所有本地队列都支持此选项。

MQGMO_ALL_SEGMENTS_AVAILABLE

仅当逻辑消息中的所有段都可用时，逻辑消息中的段才可供检索。如果队列包含缺少部分段的分段消息 (可能是因为它们在网络中已延迟并且尚未到达)，那么指定 MQGMO_ALL_SEGMENTS_AVAILABLE 将阻止检索属于不完整逻辑消息的段。但是，这些段仍构成 *CurrentQDepth* 队列属性的值；这意味着可能没有可检索的逻辑消息，即使 *CurrentQDepth* 大于零也是如此。如果没有其他可检索的消息，那么将在指定的等待时间间隔 (如果有) 到期后返回原因码 MQRC_NO_MSG_AVAILABLE。

MQGMO_ALL_SEGMENTS_AVAILABLE 的处理取决于是否还指定了 MQGMO_LOGICAL_ORDER：

- 如果同时指定了这两个选项，那么当没有当前逻辑消息时，MQGMO_ALL_SEGMENTS_AVAILABLE 仅有效。如果存在当前逻辑消息，那么将忽略 MQGMO_ALL_SEGMENTS_AVAILABLE。这意味着在按逻辑顺序处理消息时，MQGMO_ALL_SEGMENTS_AVAILABLE 可以保持开启状态。
- 如果在不带 MQGMO_LOGICAL_ORDER 的情况下指定 MQGMO_ALL_SEGMENTS_AVAILABLE，那么 MQGMO_ALL_SEGMENTS_AVAILABLE 始终具有作用。这意味着必须在从队列中除去逻辑消息中的第一个段之后关闭该选项，以便能够除去逻辑消息中的其余段。

如果未指定此选项，那么即使逻辑消息不完整，也可以检索消息段。

虽然 MQGMO_COMPLETE_MSG 和 MQGMO_ALL_SEGMENTS_AVAILABLE 都要求所有段都可用，然后才能检索它们中的任何段，但前者返回完整的消息，而后者允许逐个检索这些段。

如果为报告消息指定了 `MQGMO_ALL_SEGMENTS_AVAILABLE`，那么队列管理器将检查该队列，以查看组成完整逻辑消息的每个分段是否至少有一条报告消息。如果存在，那么满足 `MQGMO_ALL_SEGMENTS_AVAILABLE` 条件。但是，队列管理器不会检查存在的报告消息的类型，因此报告消息中可能混用了与逻辑消息段相关的报告类型。因此，`MQGMO_ALL_SEGMENTS_AVAILABLE` 的成功并不意味着 `MQGMO_COMPLETE_MSG` 将成功。如果存在为特定逻辑消息的段提供的报告类型的混合，那么必须逐个检索这些报告消息。

您可以使用任何其他 `MQGMO_*` 选项以及任何 `MQMO_*` 选项来指定 `MQGMO_ALL_SEGMENTS_AVAILABLE`。

- 在 z/OS 上，专用队列和共享队列支持此选项，但队列的索引类型必须为 `MQIT_GROUP_ID`。对于共享队列，队列映射到的 `CFSTRUCT` 对象必须处于 `CFLEVEL ($TAG1)` 或 `CFLEVEL (4)`。
- 在 AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机上，所有本地队列都支持此选项。

属性选项： 下列选项与消息的属性相关：

MQGMO_PROPERTIES_AS_Q_DEF

消息的属性 (消息描述符 (或扩展) 中包含的属性除外) 应由 `PropertyControl` 队列属性定义。如果提供了 `MsgHandle`，那么除非 `PropertyControl` 队列属性的值为 `MQPROP_FORCE_MQRFH2`，否则将忽略此选项并通过 `MsgHandle` 提供消息属性。

如果未指定属性选项，那么这是缺省操作。

MQGMO_PROPERTIES_IN_HANDLE

应通过 `MsgHandle` 提供消息的属性。如果未提供消息句柄，那么调用将失败，原因为 `MQRC_HMSG_ERROR`。

注： 如果消息稍后由未创建消息句柄的应用程序读取，那么队列管理器会将任何消息属性放入 `MQRFH2` 结构中。您可能会发现存在意外的 `MQRFH2` 头会破坏现有应用程序的行为。

MQGMO_NO_PROPERTIES

将不会检索消息的属性，但包含在消息描述符 (或扩展) 中的属性除外。如果提供了 `MsgHandle`，那么将忽略该值。

MQGMO_PROPERTIES_FORCE_MQRFH2

应该使用 `MQRFH2` 头来表示消息的属性 (消息描述符 (或扩展) 中包含的属性除外)。对于期望检索属性但无法更改为使用消息句柄的应用程序，这将提供与先前版本的兼容性。如果 `MsgHandle` 已提供，那么会将其忽略。

MQGMO_PROPERTIES_COMPATIBILITY

如果消息包含前缀为 `"mcd."`、`"jms."`、`"usr."` 或 `"mqext."` 的属性，那么所有消息属性都将通过 `MQRFH2` 头传递到应用程序。否则，将废弃除消息描述符 (或扩展) 中包含的属性之外的所有消息属性，并且应用程序再也无法访问这些属性。

缺省选项： 如果不需要所描述的任何选项，那么可以使用以下选项：

MQGMO_NONE

使用此值来指示未指定任何其他选项；所有选项均采用其缺省值。 `MQGMO_NONE` 辅助程序文档；不打算将此选项与任何其他选项一起使用，但由于其值为零，因此无法检测到此类使用。

`Options` 字段的初始值为 `MQGMO_NO_WAIT` 加号 `MQGMO_PROPERTIES_AS_Q_DEF`。

Reserved1 (MQCHAR)

这是保留字段。此字段的初始值为空白字符。如果 `Version` 小于 `MQGMO_VERSION_2`，那么将忽略此字段。

Reserved2 (MQLONG)

这是保留字段。此字段的初始值为空白字符。如果 `Version` 小于 `MQGMO_VERSION_4`，那么将忽略此字段。

ResolvedQName (MQCHAR48)

这是队列管理器设置为从中检索消息的队列的本地名称的输出字段，定义为本地队列管理器。这与用于在以下情况下打开队列的名称不同：

- 已打开别名队列 (在这种情况下，将返回已解析别名的本地队列的名称)，或者
- 已打开模型队列 (在此情况下，将返回动态本地队列的名称)。

此字段的长度由 `MQ_Q_NAME_LENGTH` 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ReturnedLength (MQLONG)

这是队列管理器在 *Buffer* 参数中设置为 `MQGET` 调用返回的消息数据的长度 (以字节计) 的输出字段。如果队列管理器不支持此功能，那么 *ReturnedLength* 将设置为值 `MQRL_UNDEFINED`。

在编码或字符集之间转换消息时，消息数据有时会更改大小。从 `MQGET` 调用返回时：

- 如果 *ReturnedLength* 不是 `MQRL_UNDEFINED`，那么返回的消息数据的字节数由 *ReturnedLength* 给出。
- 如果 *ReturnedLength* 具有值 `MQRL_UNDEFINED`，那么返回的消息数据的字节数通常由 *BufferLength* 和 *DataLength* 中的较小者提供，但如果 `MQGET` 调用完成且原因码为 `MQRC_TRUNCATED_MSG_RECEIVED`，那么可以小于此值。如果发生此情况，那么 *Buffer* 参数中的无意义字节将设置为 `null`。

定义了以下特殊值：

MQRL_UNDEFINED

未定义返回的数据的长度。

在 z/OS 上，针对 *ReturnedLength* 字段返回的值始终为 `MQRL_UNDEFINED`。

此字段的初始值为 `MQRL_UNDEFINED`。如果 *Version* 小于 `MQGMO_VERSION_3`，那么将忽略此字段。

分段 (MQCHAR)

这是一个标志，指示是否允许对检索的消息进行进一步的分段。它具有下列其中一个值：

MQSEG_禁止

不允许分段。

MQSEG_ALLOWED

允许分段。

在 z/OS 上，队列管理器始终将此字段设置为 `MQSEG_` 所在值。

这是输出字段。此字段的初始值为 `MQSEG_` 处于禁止状态。如果 *Version* 小于 `MQGMO_VERSION_2`，那么将忽略此字段。

SegmentStatus (MQCHAR)

这是一个标志，指示检索的消息是否是逻辑消息的段。它具有下列其中一个值：

MQSS_NOT_A_SEGMENT

消息不是段。

MQSS_SEGMENT

消息是一个段，但不是逻辑消息的最后一个段。

MQSS_LAST_SEGMENT

消息是逻辑消息的最后一个段。

如果逻辑消息仅由一个段组成，那么这也是返回的值。

在 z/OS 上，队列管理器始终将此字段设置为 `MQSS_NOT_A_SEGMENT`。

这是输出字段。此字段的初始值为 `MQSS_NOT_A_SEGMENT`。如果 *Version* 小于 `MQGMO_VERSION_2`，那么将忽略此字段。

Signal1 (MQLONG)

这是仅与 MQGMO_SET_SIGNAL 选项结合使用的输入字段; 它标识在消息可用时要传递的信号。

注: 此字段的数据类型和用法由环境确定; 因此, 要在不同环境之间移植的应用程序不得使用信号。

- 在 z/OS 上, 此字段必须包含事件控制块 (ECB) 的地址。在发出 MQGET 调用之前, 应用程序必须清除 ECB。在队列关闭之前, 不得释放包含 ECB 的存储器。队列管理器使用描述的其中一个信号完成代码来发布 ECB。这些完成代码以位 2 到 31 的 ECB 为单位设置, 这是 z/OS 映射宏 IHAECB 中定义的用于用户完成代码的区域。
- 在所有其他环境中, 这是保留字段; 其值不重要。

信号完成代码为:

MQEC_MSG_已到达

适合的消息已到达队列。此消息未保留给调用者; 必须发出第二个 MQGET 请求, 但另一个应用程序可能会在发出第二个请求之前检索消息。

MQEC_WAIT_INTERVAL_EXPIRED

指定的 *WaitInterval* 已到期, 没有合适的消息到达。

MQEC_WAIT_CANCEL

由于不确定原因 (例如队列管理器正在终止或正在禁用队列), 已取消等待。如果需要进一步诊断, 请重新发出请求。

MQEC_Q_MGR QUIESCING

由于队列管理器已进入停顿状态 (在 MQGET 调用上指定了 MQGMO_FAIL_IF QUIESCING), 因此已取消等待。

MQEC_CONNECTION QUIESCING

由于连接已进入停顿状态 (在 MQGET 调用上指定了 MQGMO_FAIL_IF QUIESCING), 因此已取消等待。

此字段的初始值由环境确定:

- 在 z/OS 上, 初始值为空指针。
- 在所有其他环境中, 初始值为 0。

Signal2 (MQLONG)

这是仅与 MQGMO_SET_SIGNAL 选项结合使用的输入字段。它是保留字段; 其值不重要。

此字段的初始值为 0。

StrucId (MQCHAR4)

这是结构标识。该值必须为:

MQGMO_STRUC_ID

get-message 选项结构的标识。

对于 C 编程语言, 还定义了常量 MQGMO_STRUC_ID_ARRAY; 此值与 MQGMO_STRUC_ID 相同, 但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQGMO_STRUC_ID。

Version (MQLONG)

版本是结构版本号。

值必须为以下其中一项:

MQGMO_VERSION_1

Version-1 get-message 选项结构。

此版本在所有环境中都受支持。

MQGMO_VERSION_2

Version-2 获取消息选项结构。

此版本在所有环境中都受支持。

MQGMO_VERSION_3

Version-3 get-message 选项结构。

此版本在所有环境中都受支持。

MQGMO_VERSION_4

Version-4 获取消息选项结构。

此版本在所有环境中都受支持。

仅在结构的最新版本中存在的字段在字段的描述中标识为此类字段。以下常量指定当前版本的版本号：

MQGMO_CURRENT_VERSION

当前版本的 get-message 选项结构。

这始终是一个输入字段。此字段的初始值为 MQGMO_VERSION_1。

WaitInterval (MQLONG)

这是 MQGET 调用等待合适的消息 (即, 满足 MQGET 调用的 *MsgDesc* 参数中指定的选择标准的消息) 到达的大致时间 (以毫秒为单位)。

要点: 如果有合适的消息立即可用, 那么不会等待或延迟。

有关更多详细信息, 请参阅第 364 页的『MQMD - 消息描述符』中描述的 *MsgId* 字段。) 如果经过此时间后没有合适的消息到达, 那么调用将完成, 并带有 MQCC_FAILED 和原因码 MQRC_NO_MSG_AVAILABLE。

在 z/OS 上, MQGET 调用实际等待的时间段受系统装入和工作调度注意事项影响, 并且在为 *WaitInterval* 指定的值与比 *WaitInterval* 大约 250 毫秒之间可能有所不同。

WaitInterval 与 MQGMO_WAIT 或 MQGMO_SET_SIGNAL 选项结合使用。如果未指定任何一个值, 那么将忽略该值。如果指定了其中一个值, 那么 *WaitInterval* 必须大于或等于零或以下特殊值:

MQWI_UNLIMITED

无限制的等待时间间隔。

此字段的初始值为 0。

MQGMO 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|----------------------|---|-------------------|
| <i>StrucId</i> | MQGMO_STRUC_ID | 'GMO-' |
| <i>Version</i> | MQGMO_VERSION_1 | 1 |
| <i>Options</i> | MQGMO_NO_WAIT | 0 |
| <i>WaitInterval</i> | None | 0 |
| <i>Signal1</i> | None | z/OS 上的空指针; 否则为 0 |
| <i>Signal2</i> | None | 0 |
| <i>ResolvedQName</i> | None | 空字符串或空白 |
| <i>MatchOptions</i> | MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID | 3 |
| <i>GroupStatus</i> | MQGS_NOT_IN_GROUP | '-' |
| <i>SegmentStatus</i> | MQSS_NOT_A_SEGMENT | '-' |
| <i>Segmentation</i> | MQSEG_禁止 | '-' |

表 508: MQGMO for MQGMO 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|-----------------------|----------------|------|
| <i>Reserved1</i> | None | '␣' |
| <i>MsgToken</i> | MQMTOK_NONE | Null |
| <i>ReturnedLength</i> | MQRL_UNDEFINED | -1 |
| <i>Reserved2</i> | None | '␣' |
| <i>MsgHandle</i> | MQHM_NONE | 0 |

注意:

1. 符号 ␣ 表示单个空白字符。
2. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。
3. 在 C 编程语言中，宏变量 MQGMO_DEFAULT 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值:

```
MQGMO MyGMO = {MQGMO_DEFAULT};
```

C 声明

```
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 */
};
```

- 在 z/OS 上，*Signal1* 字段声明为 PMQLONG。

COBOL 声明

```
** 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.

```

- 在 z/OS 上, *Signal1* 字段声明为 POINTER。

PL/I 声明

```

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 */

```

- 在 z/OS 上, *Signal1* 字段声明为 pointer。

High Level Assembler 声明

```

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)

```

Visual Basic 声明

```

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 信息头

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|-----------------------|---------------------|--------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>StrucLength</i> | MQIIH 结构的长度 | StrucLength |
| <i>Encoding</i> | 已保留 | 编码 |
| <i>CodedCharSetId</i> | 已保留 | CodedCharSetId |
| <i>Format</i> | MQIIH 之后数据的 MQ 格式名称 | FORMAT |
| <i>Flags</i> | 标志 | 标志 |
| <i>LTermOverride</i> | 逻辑终端覆盖 | LTermOverride |
| <i>MFSMapName</i> | 消息格式化服务映射名称 | MFSMapName |
| <i>ReplyToFormat</i> | 应答消息的 MQ 格式名称 | ReplyToFormat |
| <i>Authenticator</i> | RACF™ 密码或通行证 | Authenticator |

表 509: MQIIH 中的字段 (继续)

| 字段 | 描述 | Topic |
|-----------------------|--------|--------------------------------|
| <i>TranInstanceId</i> | 事务实例标识 | TranInstanceId |
| <i>TranState</i> | 事务状态 | TranState |
| <i>CommitMode</i> | 落实方式 | CommitMode |
| <i>SecurityScope</i> | 安全范围 | SecurityScope |
| <i>Reserved</i> | 已保留 | 已保留 |

MQIIH 概述

可用性: 所有 WebSphere MQ 系统和 WebSphere MQ 客户机。

用途: MQIIH 结构描述通过 WebSphere MQ for z/OS 发送到 IMS 网桥的消息开始时必须存在的信息。

格式名: MQFMT_IMS。

字符集和编码: 特殊条件适用于用于 MQIIH 结构和应用程序消息数据的字符集和编码:

- 连接到拥有 IMS 网桥队列的队列管理器的应用程序必须提供队列管理器的字符集和编码中的 MQIIH 结构。这是因为在这种情况下不会执行 MQIIH 结构的数据转换。
- 连接到其他队列管理器的应用程序可以提供 MQIIH 结构, 该结构位于任何受支持的字符集和编码中; 连接到拥有 IMS 网桥队列的队列管理器的接收消息通道代理将转换 MQIIH。
- 遵循 MQIIH 结构的应用程序消息数据必须采用与 MQIIH 结构相同的字符集和编码。请勿使用 MQIIH 结构中的 *CodedCharSetId* 和 *Encoding* 字段来指定应用程序消息数据的字符集和编码。

如果数据不是队列管理器支持的其中一种内置格式, 那么必须提供数据转换出口以转换应用程序消息数据。

MQIIH 的字段

MQIIH 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

鉴别符 (MQCHAR8)

这是 RACF 密码或 PassTicket。它是可选的; 如果指定了此属性, 那么它将与 MQMD 安全上下文中的用户标识配合使用, 以构建发送到 IMS 以提供安全上下文的 UTOKEN。如果未指定, 那么将在不进行验证的情况下使用用户标识。这取决于 RACF 交换机的设置, 这可能要求提供认证程序。

如果第一个字节为空或为零, 那么将忽略此值。可以使用以下特殊值:

MQIAUT_NONE

无认证。

对于 C 编程语言, 还定义了常量 MQIAUT_NONE_ARRAY; 此值与 MQIAUT_NONE 相同, 但是字符数组而不是字符串。

此字段的长度由 MQ_AUTHENTICATOR_LENGTH 指定。此字段的初始值为 MQIAUT_NONE。

CodedCharSetId (MQLONG)

这是保留字段; 其值不重要。此字段的初始值为 0。

MQIIH 结构后面的受支持结构的字符集标识与 MQIIH 结构本身的字符集标识相同, 并且取自任何前面的 MQ 头。

CommitMode (MQCHAR)

这是 IMS 落实方式。有关 IMS 落实方式的更多信息, 请参阅 OTMA 参考。值必须为以下其中一项:

MQICM_COMMIT_THEN_SEND

提交然后发送。

此方式意味着对输出进行双重排队，但区域占用时间较短。无法使用此方式运行快速路径和会话式事务。

MQICM_SEND_THEN_COMMIT

发送然后落实。

由于 MQICM_SEND_THEN_COMMIT 的落实 mpde 而启动的任何 IMS 事务都以 RESPONSE 方式运行，而不考虑如何在 IMS 系统定义中定义该事务 (TRANSACTION 宏中的 MSGTYPE 参数)。这也适用于通过事务切换启动的事务。

此字段的初始值为 MQICM_COMMIT_THEN_SEND。

Encoding (MQLONG)

这是保留字段; 其值不重要。此字段的初始值为 0。

遵循 MQIIH 结构的受支持结构的编码与 MQIIH 结构本身的编码相同，并取自任何先前的 MQ 头。

Flags (MQLONG)

标志值必须为:

MQIIH_NONE

没有标志。

MQIIH_PASS_EXPIRATION

应答消息包含:

- 与请求消息相同的到期报告选项
- 来自请求消息的剩余到期时间 (未对网桥的处理时间进行调整)

如果未设置此值，那么到期时间将设置为 无限制。

MQIIH_REPLY_FORMAT_NONE

设置 MQIIH.Format 字段。

MQIIH_IGNORE_PURG

在 OTMA 前缀中设置 TMAMIPRG 指示符，这将请求 OTMA 忽略针对 CMO 事务的 TP PCB 上的 PURG 调用。

MQIIH_CMO_REQUEST_RESPONSE

对于落实方式 0 (CMO) 事务，此标志在 OTMA 前缀中设置 TMAMHRSP 指示符。设置此指示符将请求 OTMA/IMS 生成 DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY 消息，前提是原始 IMS 应用程序未应答 IOPCB，也未将消息切换到另一个事务。

此字段的初始值为 MQIIH_NONE。

Format (MQCHAR8)

这将指定遵循 MQIIH 结构的数据的 MQ 格式名称。

在 MQPUT 或 MQPUT1 调用上，应用程序必须将此字段设置为适合于数据的值。

此字段的长度由 MQ_FORMAT_LENGTH 指定。此字段的初始值为 MQFMT_NONE。

LTermOverride (MQCHAR8)

逻辑终端覆盖，放置在 IO PCB 字段中。它是可选的; 如果未指定，那么将使用 TPIPE 名称。如果第一个字节为空或为空格，那么将忽略该值。

此字段的长度由 MQ_LTERM_OVERRIDE_LENGTH 给出。此字段的初始值为 8 个空白字符。

MFSMapName (MQCHAR8)

消息格式服务映射名称，放在 IO PCB 字段中。此字段为可选字段。在输入时，它表示 MID，在输出时，它表示 MOD。如果第一个字节为空或为空格，那么将忽略该值。

此字段的长度由 MQ_MFS_MAP_NAME_LENGTH 提供。此字段的初始值为 8 个空白字符。

ReplyTo 格式 (MQCHAR8)

这是为响应当前消息而发送的应答消息的 MQ 格式名称。此字段的长度由 MQ_FORMAT_LENGTH 指定。此字段的初始值为 MQFMT_NONE。

要使用 MQGMO_CONVERT 转换应答消息中的数据，请指定 MQIIH.replyToFormat= MQFMT_STRING 或 MQIIH.replyToFormat= MQFMT_IMS_VAR_STRING。有关使用这些字段的说明，请参阅 [第 377 页的『Format \(MQCHAR8\)』](#)。

如果在请求消息上使用缺省值 (MQIIH.replyToFormat= MQFMT_NONE)，并且使用 MQGMO_CONVERT 检索应答消息，那么不会执行数据转换。

保留 (MQCHAR)

这是保留字段；必须为空白。

SecurityScope (MQCHAR)

这指示需要 IMS 安全性处理。已定义下列值：

MQISS_CHECK

检查安全作用域: ACEE 是在控制区域中构建的，而不是在从属区域中构建的。

MQISS_FULL

完全安全作用域: 在控制区域中构建高速缓存的 ACEE，在从属区域中构建非高速缓存的 ACEE。如果使用 MQISS_FULL，请确保为其构建 ACEE 的用户标识有权访问从属区域中使用的资源。

如果未对此字段指定 MQISS_CHECK 或 MQISS_FULL，那么将采用 MQISS_CHECK。

此字段的初始值为 MQISS_CHECK。

StrucId (MQCHAR4)

这是结构标识。该值必须为：

MQIIH_STRUC_ID

IMS 信息头结构的标识。

对于 C 编程语言，还定义了常量 MQIIH_STRUC_ID_ARRAY；此值与 MQIIH_STRUC_ID 相同，但是字符数组而不是字符串。

此字段的初始值为 MQIIH_STRUC_ID。

StrucLength (MQLONG)

这是 MQIIH 结构的长度。该值必须为：

MQIIH_LENGTH_1

IMS 信息头结构的长度。

此字段的初始值为 MQIIH_LENGTH_1。

TranInstance 标识 (MQBYTE16)

这是事务实例标识。此字段由来自 IMS 的输出消息使用，因此在第一次输入时将忽略此字段。如果将 *TranState* 设置为 MQITS_IN_CONVERSATION，那么必须在下一个输入和所有后续输入中提供此内容，以使 IMS 能够将消息与正确的对话相关联。可以使用以下特殊值：

MQITII_NONE

无事务实例标识。

对于 C 编程语言，还定义了常量 MQITII_NONE_ARRAY；此值与 MQITII_NONE 相同，但是字符数组而不是字符串。

此字段的长度由 MQ_TRAN_INSTANCE_ID_LENGTH 指定。此字段的初始值为 MQITII_NONE。

TranState (MQCHAR)

这指示 IMS 对话状态。由于不存在任何对话，因此在第一次输入时将忽略此内容。在后续输入中，它指示对话是否处于活动状态。在输出时，它由 IMS 设置。值必须为以下其中一项：

MQITS_IN_CONVERSATION

在对话中。

MQITS_NOT_IN_CONVERSATION

不在对话中。

MQITS_ARCHITECTED

以架构形式返回事务状态数据。

此值仅与 IMS /DISPLAY TRAN 命令配合使用。它以 IMS 体系结构格式 (而不是字符格式) 返回事务状态数据。

此字段的初始值为 MQITS_NOT_IN_CONVERSATION。

Version (MQLONG)

这是结构版本号。该值必须为:

MQIIH_VERSION_1

IMS 信息头结构的版本号。

以下常量指定当前版本的版本号:

MQIIH_CURRENT_VERSION

当前版本的 IMS 信息头结构。

此字段的初始值为 MQIIH_VERSION_1。

MQIIH 的初始值和语言声明

| 表 510: MQIIH 的 MQIIH 中字段的初始值 | | |
|------------------------------|---------------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQIIH_STRUC_ID | 'IIH?' |
| <i>Version</i> | MQIIH_VERSION_1 | 1 |
| <i>StrucLength</i> | MQIIH_LENGTH_1 | 84 |
| <i>Encoding</i> | None | 0 |
| <i>CodedCharSetId</i> | None | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQIIH_NONE | 0 |
| <i>LTermOverride</i> | None | 空白 |
| <i>MFSMapName</i> | None | 空白 |
| <i>ReplyToFormat</i> | MQFMT_NONE | 空白 |
| <i>Authenticator</i> | MQIAUT_NONE | 空白 |
| <i>TranInstanceId</i> | MQITII_NONE | Null |
| <i>TranState</i> | MQITS_NOT_IN_CONVERSATION | '?' |
| <i>CommitMode</i> | MQICM_COMMIT_THEN_SEND | '0' |
| <i>SecurityScope</i> | MQISS_CHECK | 'C' |
| <i>Reserved</i> | None | '?' |

表 510: MQIIH 的 MQIIH 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|--|-------|-----|
| 注意: | | |
| 1. 符号? 表示单个空白字符。 2. 在 C 编程语言中, 宏变量 MQIIH_DEFAULT 包含以上列出的值。 它可以通过以下方式用于为结构中的字段提供初始值: | | |
| <pre>MQIIH MyIIH = {MQIIH_DEFAULT};</pre> | | |

C 声明

```
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   MFMapName;       /* 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 声明

```
** 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.
```

PL/I 声明

```
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 */
```

High Level Assembler 声明

```
MQIIH          DSECT
MQIIH_STRUCID  DS   CL4   Structure identifier
MQIIH_VERSION  DS   F     Structure version number
MQIIH_STRUCLNGTH DS  F     Length of MQIIH structure
MQIIH_ENCODING DS   F     Reserved
MQIIH_CODEDCCHARSETID 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)
```

Visual Basic 声明

```
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-查询消息属性选项

下表汇总了结构中的字段。MQIMPO 结构-查询消息属性选项

| 字段 | 描述 | Topic |
|--------------------------|------------------|-----------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 控制 MQINQMP 操作的选项 | 选项 |
| <i>RequestedEncoding</i> | 要将查询的属性转换为的编码 | RequestedEncoding |
| <i>RequestedCCSID</i> | 所查询属性的字符集 | RequestedCCSID |
| <i>ReturnedEncoding</i> | 返回值的编码 | ReturnedEncoding |
| <i>ReturnedCCSID</i> | 返回值的字符集 | ReturnedCCSID |
| <i>Reserved1</i> | 保留字段 | ReturnedCCSID |
| <i>ReturnedName</i> | 查询的属性的名称 | ReturnedName |
| <i>TypeString</i> | 属性的数据类型的字符串表示 | TypeString |

MQIMPO 概述

查询消息属性选项结构。

可用性: 所有 WebSphere MQ 系统和 WebSphere MQ 客户机。

用途:MQIMPO 结构允许应用程序指定用于控制如何查询消息属性的选项。该结构是 MQINQMP 调用上的输入参数。

字符集和编码:MQIMPO 中的数据必须在应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQIMPO 的字段

查询消息属性选项结构-字段

MQIMPO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

选项 (MQLONG)

查询消息属性选项结构-"选项" 字段

以下选项控制 MQINQMP 的操作。您可以指定其中一个或多个选项, 如果需要多个选项, 那么值可以是:

- 一起添加 (请勿多次添加同一常量), 或者
- 通过按位“或”运算组合在一起 (如果编程语言支持位运算)。

将记录无效选项的组合; 所有其他组合都有效。

值数据选项: 从消息中检索属性时, 以下选项与值数据的处理相关。

MQIMPO_CONVERT_VALUE

此选项请求转换属性值以符合在 MQINQMP 调用返回 *Value* 区域中的属性值之前指定的 *RequestedCCSID* 和 *RequestedEncoding* 值。

- 如果转换成功, 那么 *ReturnedCCSID* 和 *ReturnedEncoding* 字段将设置为与从 MQINQMP 调用返回时的 *RequestedCCSID* 和 *RequestedEncoding* 相同。
- 如果转换失败, 但 MQINQMP 调用在未发生错误的情况下完成, 那么将返回未转换的属性值。

如果该属性是字符串, 那么 *ReturnedCCSID* 和 *ReturnedEncoding* 字段将设置为未转换字符串的字符集和编码。

在这种情况下，完成代码为 MQCC_WARNING，原因码为 MQRC_PROP_VALUE_NOT_CONVERTED。属性光标将高级到返回的属性。

如果属性值在转换期间扩展，并且超过 *Value* 参数的大小，那么将返回未转换的值，完成代码为 MQCC_FAILED; 原因码设置为 MQRC_PROPERTY_VALUE_TOO_BIG。

MQINQMP 调用的 *DataLength* 参数返回属性值将转换为的长度，以便允许应用程序确定容纳转换后的属性值所需的缓冲区大小。属性光标未更改。

此选项还请求：

- 如果属性名称包含通配符，并且
- 使用返回的名称的地址或偏移量初始化 *ReturnedName* 字段。

那么返回的名称将转换为符合 *RequestedCCSID* 和 *RequestedEncoding* 值。

- 如果转换成功，那么 *ReturnedName* 的 *VSCCSID* 字段和返回名称的编码将设置为输入值 *RequestedCCSID* 和 *RequestedEncoding*。
- 如果转换失败，但 MQINQMP 调用在未发生错误或警告的情况下完成，那么将不转换返回的名称。在这种情况下，完成代码为 MQCC_WARNING，原因码为 MQRC_PROP_NAME_NOT_CONVERTED。

属性光标将高级到返回的属性。如果未转换值和名称，那么将返回 MQRC_PROP_VALUE_NOT_CONVERTED。

如果返回的名称在转换期间扩展，并且超出 *RequestedName* 的 *VSBuFSIZE* 字段的大小，那么返回的字符串将保持未转换状态，完成代码为 MQCC_FAILED，原因码设置为 MQRC_PROPERTY_NAME_TOO_BIG。

MQCHARV 结构的 *VSLength* 字段返回属性值将转换为的长度，以便允许应用程序确定容纳转换后的属性值所需的缓冲区大小。属性光标未更改。

MQIMPO_CONVERT_TYPE

此选项请求将属性值从其当前数据类型转换为 MQINQMP 调用的 *Type* 参数上指定的数据类型。

- 如果转换成功，那么 *Type* 参数在返回 MQINQMP 调用时保持不变。
- 如果转换失败，但 MQINQMP 调用在未发生错误的情况下完成，那么调用将失败，原因为 MQRC_PROP_CONV_NOT_SUPPORTED。属性光标未更改。

如果数据类型的转换导致值在转换期间扩展，并且转换后的值超过 *Value* 参数的大小，那么将返回未转换的值，完成代码为 MQCC_FAILED，原因码设置为 MQRC_PROPERTY_VALUE_TOO_BIG。

MQINQMP 调用的 *DataLength* 参数返回属性值将转换为的长度，以便允许应用程序确定容纳转换后的属性值所需的缓冲区大小。属性光标未更改。

如果 MQINQMP 调用的 *Type* 参数值无效，那么调用将失败，原因为 MQRC_PROPERTY_TYPE_ERROR。

如果不支持所请求的数据类型转换，那么调用将失败，原因为 MQRC_PROP_CONV_NOT_SUPPORTED。支持以下数据类型转换：

| 属性数据类型 | 受支持的目标数据类型 |
|--------------------|--|
| 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 和 MQTYPE_INT64 |
| MQTYPE_INT16 | MQTYPE_STRING, MQTYPE_INT32 和 MQTYPE_INT64 |
| MQTYPE_INT32 | MQTYPE_STRING 和 MQTYPE_INT64 |
| MQTYPE_INT64 | MQTYPE_STRING |
| MQTYPE_FLOAT32 | MQTYPE_STRING, MQTYPE_FLOAT64 |

| 属性数据类型 | 受支持的目标数据类型 |
|----------------|---|
| MQTYPE_FLOAT64 | MQTYPE_STRING |
| MQTYPE_STRING | MQTYPE_BOOLEAN, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64, MQTYPE_FLOAT32, MQTYPE_FLOAT64 |
| MQTYPE_NULL | None |

用于管理受支持转换的一般规则如下：

- 可以将数字属性值从一种数据类型转换为另一种数据类型，前提是在转换期间不会丢失任何数据。
例如，数据类型为 MQTYPE_INT32 的属性值可以转换为数据类型为 MQTYPE_INT64 的值，但不能转换为数据类型为 MQTYPE_INT16 的值。
- 可以将任何数据类型的属性值转换为字符串。
- 可以将字符串属性值转换为任何其他数据类型，前提是已针对转换正确格式化了该字符串。如果应用程序尝试转换未正确格式化的字符串属性值，那么 WebSphere MQ 将返回原因码 MQRC_PROP_NUMBER_FORMAT_ERROR。
- 如果应用程序尝试进行不受支持的转换，那么 WebSphere MQ 将返回原因码 MQRC_PROP_CONV_NOT_SUPPORTED。

将属性值从一种数据类型转换为另一种数据类型的具体规则如下：

- 将 MQTYPE_BOOLEAN 属性值转换为字符串时，值 TRUE 将转换为字符串 "TRUE"，而值 false 将转换为字符串 "FALSE"。
- 将 MQTYPE_BOOLEAN 属性值转换为数字数据类型时，值 TRUE 将转换为 1，而值 FALSE 将转换为零。
- 将字符串属性值转换为 MQTYPE_BOOLEAN 值时，字符串 "TRUE" 或 "1" 将转换为 TRUE，而字符串 "FALSE" 或 "\$TAG2" 将转换为 FALSE。

请注意，术语 "TRUE" 和 "FALSE" 不区分大小写。

无法转换任何其他字符串；WebSphere MQ 返回原因码 MQRC_PROP_NUMBER_FORMAT_ERROR。

- 将字符串属性值转换为数据类型为 MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 或 MQTYPE_INT64 的值时，该字符串必须具有以下格式：

```
[blanks][sign]digits
```

该字符串各个组成部分的含义如下：

blanks

可选前导空白字符

sign

可选的加号 (+) 或减号 (-) 字符。

digits

连续的数字字符序列 (0-9)。必须至少存在一个数字字符。

在数字字符序列之后，该字符串可包含其他非数字字符，但是在到达这些字符中的第一个字符后会立即停止转换。假设该字符串表示十进制整数。

如果字符串的格式不正确，WebSphere MQ 将返回原因码 MQRC_PROP_NUMBER_FORMAT_ERROR。

- 将字符串属性值转换为数据类型为 MQTYPE_FLOAT32 或 MQTYPE_FLOAT64 的值时，该字符串必须具有以下格式：

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

该字符串各个组成部分的含义如下：

blanks

可选前导空白字符

sign

可选的加号 (+) 或减号 (-) 字符。

digits

连续的数字字符序列 (0-9)。必须至少存在一个数字字符。

e_char

一个指数字符，即 "E" 或 "e"。

e_sign

指数的可选加号 (+) 或减号 (-) 字符。

e_digits

该指数的连续数字字符序列 (0-9)。如果该字符串包含指数字符，那么必须至少存在一个数字字符。

在数字字符序列或表示指数的可选字符之后，该字符串可包含其他非数字字符，但是在到达这些字符中的第一个字符后会立即停止转换。假设该字符串表示十进制浮点数，指数幂为 10。

如果字符串的格式不正确，WebSphere MQ 将返回原因码 MQRC_PROP_NUMBER_FORMAT_ERROR。

- 将数字属性值转换为字符串时，该值将转换为该值作为十进制数字的字符串表示，而不是包含该值的 ASCII 字符的字符串。例如，整数 65 转换为字符串 "65"，而不是字符串 "A"。
- 将字节字符串属性值转换为字符串时，每个字节将转换为表示字节的两个十六进制字符。例如，字节数组 {0xF1, 0x12, 0x00, 0xFF} 将转换为字符串 "F11200FF"。

MQIMPO_QUERY_LENGTH

查询属性值的类型和长度。在 MQINQMP 调用的 *DataLength* 参数中返回长度。未返回属性值。

如果指定了 *ReturnedName* 缓冲区，那么将使用属性名称的长度填充 MQCHARV 结构的 *VSLength* 字段。未返回属性名称。

迭代选项: 以下选项与使用带有通配符的名称对属性进行迭代相关

MQIMPO_INQ_FIRST

查询与指定名称匹配的第一个属性。在此调用之后，将在返回的属性上建立光标。

这是缺省值。

MQIMPO_INQ_PROP_UNDER_CURSOR 选项随后可与 MQINQMP 调用配合使用 (如果需要)，以再次查询同一属性。

请注意，只有一个属性光标; 因此，如果 MQINQMP 调用中指定的属性名更改，那么将重置光标。

对于以下任一选项，此选项无效:

MQIMPO_INQ_NEXT
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_NEXT

查询与指定名称匹配的下一个属性，然后从属性光标继续搜索。光标将前进到返回的属性。

如果这是指定名称的第一个 MQINQMP 调用，那么将返回与指定名称匹配的第三个属性。

MQIMPO_INQ_PROP_UNDER_CURSOR 选项随后可与 MQINQMP 调用配合使用 (如果需要)，以再次查询同一属性。

如果已删除光标下的属性，那么 MQINQMP 将返回已删除的属性之后的下一个匹配属性。

如果添加了与通配符匹配的属性，那么在迭代进行期间，该属性可能在迭代完成期间返回，也可能不会返回。使用 MQIMPO_INQ_FIRST 重新启动迭代后，将返回此属性。

当迭代正在进行时，与已删除的通配符匹配的属性不会在其删除后返回。

对于以下任一选项，此选项无效:

MQIMPO_INQ_FIRST

MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_PROP_UNDER_CURSOR

检索属性光标指向的属性值。属性光标指向的属性是上次使用 MQIMPO_INQ_FIRST 或 MQIMPO_INQ_NEXT 选项查询的属性。

当复用消息句柄时，在 MQGET 调用的 MQGMO 的 *MsgHandle* 字段中指定消息句柄时，或者在 MQPUT 调用的 MQPMO 结构的 *OriginalMsgHandle* 或 *NewMsgHandle* 字段中指定消息句柄时，将重置属性光标。

如果在尚未建立属性光标时使用此选项，或者如果已删除属性光标指向的属性，那么调用将失败，并返回完成代码 MQCC_FAILED 和原因 MQRC_PROPERTY_NOT_AVAILABLE。

对于以下任一选项，此选项无效：

MQIMPO_INQ_FIRST

MQIMPO_INQ_NEXT

如果不需要先前描述的任何选项，那么可以使用以下选项：

MQIMPO_NONE

使用此值来指示未指定任何其他选项；所有选项均采用其缺省值。

MQIMPO_NONE 帮助程序文档；不打算将此选项与任何其他选项一起使用，但由于其值为零，因此无法检测到此类使用。

这始终是一个输入字段。此字段的初始值为 MQIMPO_INQ_FIRST。

RequestedCCSID (MQLONG)

查询消息属性选项结构- RequestedCCSID 字段

要将查询的属性值转换为的字符集(如果该值是字符串)。这也是指定 MQIMPO_CONVERT_VALUE 或 MQIMPO_CONVERT_TYPE 时要将 *ReturnedName* 转换为的字符集。

此字段的初始值为 MQCCSI_APPL。

RequestedEncoding (MQLONG)

查询消息属性选项结构- RequestedEncoding 字段

这是在指定 MQIMPO_CONVERT_VALUE 或 MQIMPO_CONVERT_TYPE 时要将查询的属性值转换为的编码。

此字段的初始值为 MQENC_NATIVE。

Reserved1 (MQCHAR)

这是保留字段。此字段的初始值为空白字符(4 字节字段)。

ReturnedCCSID (MQLONG)

查询消息属性选项结构- ReturnedCCSID 字段

在输出时，如果 MQINQMP 调用的 *Type* 参数为 MQTYPE_STRING，那么这是返回的值的字符集。

如果指定了 MQIMPO_CONVERT_VALUE 选项并且转换成功，那么返回时 *ReturnedCCSID* 字段的值与传入的值相同。

此字段的初始值为零。

ReturnedEncoding (MQLONG)

查询消息属性选项结构- ReturnedEncoding 字段

在输出时，这是返回的值的编码。

如果指定了 MQIMPO_CONVERT_VALUE 选项并且转换成功，那么返回时 *ReturnedEncoding* 字段的值与传入的值相同。

此字段的初始值为 MQENC_NATIVE。

ReturnedName (MQCHARV)

查询消息属性选项结构- ReturnedName 字段

查询的属性的实际名称。

在输入时，可以使用 MQCHARV 结构的 *VSPtr* 或 *VSOffset* 字段来传递字符串缓冲区。字符串缓冲区的长度是使用 MQCHARV 结构的 *VSBufsize* 字段指定的。

从 MQINQMP 调用返回时，将使用查询的属性的名称完成字符串缓冲区，前提是字符串缓冲区的长度足以完全包含该名称。MQCHARV 结构的 *VSLength* 字段中填充了属性名称的长度。MQCHARV 结构的 *VSCCSID* 字段将填充以指示返回的名称的字符集，无论该名称的转换是否失败。

这是输入/输出字段。此字段的初始值为 MQCHARV_DEFAULT。

StrucId (MQCHAR4)

查询消息属性选项结构- StrucId 字段

这是结构标识。该值必须为：

MQIMPO_STRUC_ID

查询消息属性选项结构的标识。

对于 C 编程语言，还定义了常量 MQIMPO_STRUC_ID_ARRAY；此值与 MQIMPO_STRUC_ID 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQIMPO_STRUC_ID。

TypeString (MQCHAR8)

查询消息属性选项结构- TypeString 字段

属性的数据类型的字符串表示。

如果在 MQRFH2 头中指定了该属性，并且无法识别 MQRFH2 dt 属性，那么可以使用此字段来确定该属性的数据类型。TypeString 以编码字符集 1208 (UTF-8) 返回，是无法识别的属性 dt 属性值的前 8 个字节

这始终是输出字段。此字段的初始值是 C 编程语言中的空字符串，以及其他编程语言中的 8 空白字符。

Version (MQLONG)

查询消息属性选项结构-"版本" 字段

这是结构版本号。该值必须为：

MQIMPO_VERSION_1

查询消息属性选项结构的版本号。

以下常量指定当前版本的版本号：

MQIMPO_CURRENT_VERSION

当前版本的查询消息属性选项结构。

这始终是一个输入字段。此字段的初始值为 MQIMPO_VERSION_1。

MQIMPO 的初始值和语言声明

查询消息属性选项结构-初始值

表 512: MQIPMO 中字段的初始值

| 字段名称 | 常量的名称 | 常量值 |
|--------------------------|------------------|--------|
| <i>StrucId</i> | MQIMPO_STRUC_ID | 'IMPO' |
| <i>Version</i> | MQIMPO_VERSION_1 | 1 |
| <i>Options</i> | MQIMPO_INQ_FIRST | |
| <i>RequestedEncoding</i> | MQENC_NATIVE | |
| <i>RequestedCCSID</i> | MQCCSI_APPL | |
| <i>ReturnedEncoding</i> | MQENC_NATIVE | |
| <i>ReturnedCCSID</i> | 0 | |
| <i>Reserved1</i> | 0 | |
| <i>ReturnedName</i> | MQCHARV_DEFAULT | |
| <i>TypeString</i> | 空字符串或空白 | |

注意:

1. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。
2. 在 C 编程语言中，宏变量 MQIMPO_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQIMPO MyIMPO = {MQIMPO_DEFAULT};
```

C 声明

查询消息属性选项结构-C 语言声明

```
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 声明

查询消息属性选项结构-COBOL 语言声明

```
** 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.

```

PL/I 声明

查询消息属性选项结构-PL/I 语言声明

```

dcl
  1 MQIMPO based,
  3 StructId          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 */

```

High Level Assembler 声明

查询消息属性选项结构-汇编程序语言声明

```

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 - 消息描述符

下表汇总了结构中的字段。

表 513: MQMD 中的字段

| 字段 | 描述 | Topic |
|--|-------------------|-----------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Report</i> | 报告消息的选项 | 报告 |
| <i>MsgType</i> | 消息类型 | MsgType |
| <i>Expiry</i> | 消息生命周期 | MQMD-到期字段 |
| <i>Feedback</i> | 反馈或原因码 | MQMD-反馈字段 |
| <i>Encoding</i> | 消息数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | 消息数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | 消息数据的格式名称 | 格式 |
| <i>Priority</i> | 消息优先级 | 优先级 |
| <i>Persistence</i> | 消息持久性 | 持久性 |
| <i>MsgId</i> | 消息标识 | MQMD- MsgId 字段 |
| <i>CorrelId</i> | 相关标识 | CorrelId |
| <i>BackoutCount</i> | 回退计数器 | BackoutCount |
| <i>ReplyToQ</i> | 应答队列的名称 | ReplyToQ |
| <i>ReplyToQMgr</i> | 应答队列管理器的名称 | ReplyToQMgr |
| <i>UserIdentifier</i> | 用户标识 | UserIdentifier |
| <i>AccountingToken</i> | 记帐标记 | AccountingToken |
| <i>ApplIdentityData</i> | 与身份有关的应用程序数据 | ApplIdentityData |
| <i>PutApplType</i> | 放置消息的应用程序的类型 | PutApplType |
| <i>PutApplName</i> | 放置消息的应用程序的名称 | PutApplName |
| <i>PutDate</i> | 消息的放置日期 | PutDate |
| <i>PutTime</i> | 消息放置的时间 | PutTime |
| <i>ApplOriginData</i> | 与源有关的应用程序数据 | ApplOriginData |
| 注: 如果 <i>Version</i> 小于 MQMD_VERSION_2, 那么将忽略其余字段。 | | |
| <i>GroupId</i> | 组标识 | GroupId |
| <i>MsgSeqNumber</i> | 组内逻辑消息的序列号 | MsgSeqNumber |
| <i>Offset</i> | 从逻辑消息开始的物理消息数据偏移量 | 偏移量 |
| <i>MsgFlags</i> | 消息标志 | MQMD- MsgFlags 字段 |
| <i>OriginalLength</i> | 原始消息的长度 | OriginalLength |

MQMD 概述

可用性: 所有 WebSphere MQ 系统, 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: MQMD 结构包含在发送和接收应用程序之间传输消息时伴随应用程序数据的控制信息。此结构是 MQGET, MQPUT 和 MQPUT1 调用上的输入/输出参数。

版本:MQMD的当前版本为 MQMD_VERSION_2。要在多个环境之间可移植的应用程序必须确保在所有相关环境中支持所需的 MQMD 版本。仅在结构的最新版本中存在的字段在随后的描述中标识为此类字段。

为受支持的编程语言提供的头 COPY 和 INCLUDE 文件包含环境支持的最新版本的 MQMD，但 *Version* 字段的初始值设置为 MQMD_VERSION_1。要使用 version-1 结构中不存在的字段，应用程序必须将 *Version* 字段设置为所需版本的版本号。

名称为 MQMD1 的 version-1 结构的声明可用。

字符集和编码:MQMD 中的数据必须采用本地队列管理器的字符集和编码; 这些数据由 *CodedCharSetId* 队列管理器属性和 MQENC_NATIVE 提供。但是，如果应用程序作为 WebSphere MQ MQI 客户机运行，那么该结构必须采用客户机的字符集和编码。

如果发送和接收队列管理器使用不同的字符集或编码，那么将自动转换 MQMD 中的数据。应用程序不需要转换 MQMD。

使用不同版本的 MQMD: version-2 MQMD 等同于使用 version-1 MQMD 并以 MQMDE 结构作为消息数据的前缀。但是，如果 MQMDE 结构中的所有字段都具有其缺省值，那么可以省略 MQMDE。使用 version-1 MQMD 和 MQMDE，如下所述:

- 在 MQPUT 和 MQPUT1 调用上，如果应用程序提供了 version-1 MQMD，那么应用程序可以选择使用 MQMDE 作为消息数据的前缀，将 MQMD 中的 *Format* 字段设置为 MQFMT_MD_EXTENSION 以指示存在 MQMDE。如果应用程序未提供 MQMDE，那么队列管理器将为 MQMDE 中的字段采用缺省值。

注: version-2 MQMD 中存在但不存在 version-1 MQMD 的多个字段是 MQPUT 和 MQPUT1 调用上的输入/输出字段。但是，在 MQPUT 和 MQPUT1 调用的输出中，队列管理器不会返回 MQMDE 中等效字段中的任何值; 如果应用程序需要这些输出值，那么它必须使用 version-2 MQMD。

- 在 MQGET 调用上，如果应用程序提供 version-1 MQMD，那么队列管理器会将使用 MQMDE 返回的消息作为前缀，但前提是 MQMDE 中的一个或多个字段具有非缺省值。MQMD 中的 *Format* 字段将具有值 MQFMT_MD_EXTENSION，以指示存在 MQMDE。

队列管理器用于 MQMDE 中的字段的缺省值与这些字段的初始值相同，如 第 410 页的表 518 中所示。

当消息位于传输队列上时，MQMD 中的某些字段将设置为特定值; 请参阅 第 540 页的『MQXQH-传输队列头』以获取详细信息。

消息上下文:MQMD 中的某些字段包含消息上下文。有两种类型的消息上下文: 身份上下文和源上下文。具体方式通常为:

- 身份上下文与最初放置消息的应用程序相关
- 源上下文与最近放入消息的应用程序相关。

这两个应用程序可以是同一个应用程序，但也可以是不同的应用程序 (例如，当消息从一个应用程序转发到另一个应用程序时)。

虽然身份和源上下文通常具有描述的含义，但 MQMD 中两种类型的上下文字段的内容都取决于放入消息时指定的 MQPMO_*_CONTEXT 选项。因此，身份上下文不一定与最初放置消息的应用程序相关，而源上下文不一定与最近放置消息的应用程序相关; 这取决于应用程序套件的设计。

消息通道代理程序 (MCA) 从不改变消息上下文。从远程队列管理器接收消息的 MCA 在 MQPUT 或 MQPUT1 调用上使用上下文选项 MQPMO_SET_ALL_CONTEXT。这允许接收 MCA 保留与来自发送 MCA 的消息一起传递的消息上下文。但是，结果是源上下文与发送和接收消息的任何 MCA 都不相关。源上下文是指放置消息的较早应用程序。如果所有中间应用程序都已传递消息上下文，那么源上下文将引用源应用程序本身。

在描述中，上下文字段被描述为如同先前所描述的那样使用。有关消息上下文的更多信息，请参阅 [消息上下文](#)。

MQMD 的字段

MQMD 结构包含以下字段; 这些字段按 **字母顺序**进行描述:

| 表 514: MQMD 中的字段 | | |
|------------------|------|-------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |

表 514: MQMD 中的字段 (继续)

| 字段 | 描述 | Topic |
|--|-------------------|-----------------------------------|
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Report</i> | 报告消息的选项 | 报告 |
| <i>MsgType</i> | 消息类型 | MsgType |
| <i>Expiry</i> | 消息生命周期 | MQMD-到期字段 |
| <i>Feedback</i> | 反馈或原因码 | MQMD-反馈字段 |
| <i>Encoding</i> | 消息数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | 消息数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | 消息数据的格式名称 | 格式 |
| <i>Priority</i> | 消息优先级 | 优先级 |
| <i>Persistence</i> | 消息持久性 | 持久性 |
| <i>MsgId</i> | 消息标识 | MQMD- MsgId 字段 |
| <i>CorrelId</i> | 相关标识 | CorrelId |
| <i>BackoutCount</i> | 回退计数器 | BackoutCount |
| <i>ReplyToQ</i> | 应答队列的名称 | ReplyToQ |
| <i>ReplyToQMgr</i> | 应答队列管理器的名称 | ReplyToQMgr |
| <i>UserIdentifier</i> | 用户标识 | UserIdentifier |
| <i>AccountingToken</i> | 记帐标记 | AccountingToken |
| <i>ApplIdentityData</i> | 与身份有关的应用程序数据 | ApplIdentityData |
| <i>PutApplType</i> | 放置消息的应用程序的类型 | PutApplType |
| <i>PutApplName</i> | 放置消息的应用程序的名称 | PutApplName |
| <i>PutDate</i> | 消息的放置日期 | PutDate |
| <i>PutTime</i> | 消息放置的时间 | PutTime |
| <i>ApplOriginData</i> | 与源有关的应用程序数据 | ApplOriginData |
| 注: 如果 <i>Version</i> 小于 MQMD_VERSION_2, 那么将忽略其余字段。 | | |
| <i>GroupId</i> | 组标识 | GroupId |
| <i>MsgSeqNumber</i> | 组内逻辑消息的序列号 | MsgSeqNumber |
| <i>Offset</i> | 从逻辑消息开始的物理消息数据偏移量 | 偏移量 |
| <i>MsgFlags</i> | 消息标志 | MQMD- MsgFlags 字段 |
| <i>OriginalLength</i> | 原始消息的长度 | OriginalLength |

AccountingToken (MQBYTE32)

这是记帐令牌, 是消息的 **身份上下文** 的一部分。有关消息上下文的更多信息, 请参阅第 365 页的『MQMD 概述』; 另请参阅消息上下文。

AccountingToken 允许应用程序对由于消息而完成的工作进行相应收费。队列管理器将此信息视为位字符串, 并且不检查其内容。

队列管理器生成此信息, 如下所示:

- 字段的第一个字节设置为随后的字节中存在的记帐信息的长度; 此长度在范围 0 到 30 之间, 并以二进制整数形式存储在第一个字节中。
- 第二个字节和后续字节 (由长度字段指定) 设置为适合于环境的记帐信息。
 - 在 z/OS 上, 记帐信息设置为:
 - 对于 z/OS 批处理, 来自 JES JOB 卡或 EXEC 卡中的 JES ACCT 语句的记帐信息 (逗号分隔符将更改为 X'FF ')。如果需要, 此信息将截断为 31 个字节。
 - 对于 TSO, 用户的帐号。
 - 对于 CICS, LU 6.2 工作单元标识 (UEPUOWDS) (26 字节)。
 - 对于 IMS, 8 字符 PSB 名称与 16 个字符的 IMS 恢复令牌并置。
 - 在 IBM i 上, 会将记帐信息设置为作业的记帐代码。
 - 在 UNIX 系统上, 记帐信息设置为以 ASCII 字符表示的数字用户标识。
 - 在 Windows 上, 记帐信息设置为压缩格式的 Windows 安全标识 (SID)。SID 唯一地标识存储在 *UserIdentifier* 字段中的用户标识。当 SID 存储在 *AccountingToken* 字段中时, 将省略 6 字节的 "标识权限" (位于 SID 的第三个字节和后续字节中)。例如, 如果 Windows SID 的长度为 28 个字节, 那么会在 *AccountingToken* 字段中存储 22 个字节的 SID 信息。
- 记帐字段的最后一个字节 (字节 32) 设置为记帐令牌类型 (在本例中为 MQACTT_NT_SECURITY_ID, x '0b'):

MQACTT_CICS_LUOW_ID

CICS LUOW 标识。

MQACTT_NT_SECURITY_ID

Windows 安全标识。

MQACTT_OS400_ACCOUNT_TOKEN

IBM i 记帐标记。

MQACTT_UNIX_NUMERIC_ID

UNIX 系统数字标识。

MQACTT_USER

用户定义的记帐令牌。

MQACTT_UNKNOWN

未知的记帐令牌类型。

仅在以下环境中将记帐令牌类型设置为显式值: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。在其他环境中, 帐户令牌类型设置为值 MQACTT_UNKNOWN。在这些环境中, 使用 *PutApplType* 字段来推断接收的记帐令牌的类型。

- 所有其他字节都设置为二进制零。

对于 MQPUT 和 MQPUT1 调用, 如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_IDENTITY_CONTEXT 或 MQPMO_SET_ALL_CONTEXT, 那么这是输入/输出字段。如果既未指定 MQPMO_SET_IDENTITY_CONTEXT, 也未指定 MQPMO_SET_ALL_CONTEXT, 那么此字段在输入时将被忽略, 并且是仅输出字段。有关消息上下文的更多信息, 请参阅 [消息上下文](#)。

成功完成 MQPUT 或 MQPUT1 调用后, 此字段包含随消息一起传输的 *AccountingToken* (如果将其放入队列)。如果保留消息 (请参阅第 438 页的『MQPMO 选项 (MQLONG)』中对 MQPMO_RETAIN 的描述以获取有关保留发布的更多详细信息), 那么这将是随消息一起保留的 *AccountingToken* 的值, 但是在将消息作为发布发送给订户时, 不会将其用作 *AccountingToken*, 因为它们提供了一个值来覆盖发送给订户的所有发布中的 *AccountingToken*。如果消息没有上下文, 那么该字段完全为二进制零。

这是 MQGET 调用的输出字段。

此字段不受基于队列管理器字符集的任何转换; 该字段被视为位字符串, 而不被视为字符串。

队列管理器对此字段中的信息不执行任何操作。如果应用程序想要将该信息用于记帐目的, 那么它必须解释该信息。

可以将以下特殊值用于 *AccountingToken* 字段:

MQACT_NONE

未指定记帐标记。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQACT_NONE_ARRAY; 此值与 MQACT_NONE 相同，但是字符数组而不是字符串。

此字段的长度由 MQ_ACCOUNTING_TOKEN_LENGTH 提供。此字段的初始值为 MQACT_NONE。

ApplIdentity 数据 (MQCHAR32)

这是消息的 **身份上下文** 的一部分。有关消息上下文的更多信息，请参阅 [第 365 页的『MQMD 概述』](#) 和 [消息上下文](#)。

ApplIdentityData 是由应用程序套件定义的信息，可用于提供有关消息或其发起方的其他信息。队列管理器将此信息视为字符数据，但不定义其格式。当队列管理器生成此信息时，它完全为空白。

对于 MQPUT 和 MQPUT1 调用，如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_IDENTITY_CONTEXT 或 MQPMO_SET_ALL_CONTEXT，那么这是输入/输出字段。如果存在空字符，那么队列管理器会将空字符和以下任何字符转换为空白。如果既未指定 MQPMO_SET_IDENTITY_CONTEXT，也未指定 MQPMO_SET_ALL_CONTEXT，那么此字段在输入时将被忽略，并且是仅输出字段。有关消息上下文的更多信息，请参阅 [消息上下文](#)。

成功完成 MQPUT 或 MQPUT1 调用后，此字段包含随消息一起传输的 *ApplIdentityData* (如果将其放入队列)。如果保留消息，那么这将是 *ApplIdentityData* 的值 (请参阅 MQPMO_RETAIN 的描述以获取有关保留发布的更多详细信息)，但在将消息作为发布内容发送给订户时，不会将此值用作 *ApplIdentityData*，因为它们提供的值将覆盖发送给订户的所有发布内容中的 *ApplIdentityData*。如果消息没有上下文，那么该字段完全为空白。

这是 MQGET 调用的输出字段。此字段的长度由 MQ_APPL_IDENTITY_DATA_LENGTH 给出。此字段的初始值是 C 中的空字符串，在其他编程语言中为 32 个空白字符。

ApplOrigin 数据 (MQCHAR4)

这是消息的 **源上下文** 的一部分。有关消息上下文的更多信息，请参阅 [第 365 页的『MQMD 概述』](#) 和 [消息上下文](#)。

ApplOriginData 是应用程序套件定义的信息，可用于提供有关消息源的其他信息。例如，它可以由使用适当用户权限运行的应用程序设置，以指示身份数据是否可信。

队列管理器将此信息视为字符数据，但不定义其格式。当队列管理器生成此信息时，它完全为空白。

对于 MQPUT 和 MQPUT1 调用，如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_ALL_CONTEXT，那么这是输入/输出字段。将废弃字段中空字符后面的任何信息。队列管理器将空字符和任何后续字符转换为空白。如果未指定 MQPMO_SET_ALL_CONTEXT，那么此字段将在输入时被忽略，并且是仅输出字段。

这是 MQGET 调用的输出字段。此字段的长度由 MQ_APPL_ORIGIN_DATA_LENGTH 提供。此字段的初始值是 C 中的空字符串，以及其他编程语言中的 4 个空白字符。

发布消息时，虽然设置了 *ApplOriginData*，但它在接收的预订中为空。

BackoutCount (MQLONG)

这是 MQGET 调用先前作为工作单元一部分返回并随后回退消息的次数的计数。它可帮助应用程序检测基于消息内容的处理错误。此计数不包括指定任何 MQGMO_BROWSE_* 选项的 MQGET 调用。

此计数的准确性受 *HardenGetBackout* 队列属性影响; 请参阅 [第 731 页的『队列的属性』](#)。

在 z/OS 上，值 255 表示消息已回退 255 次或更多次; 返回的值从不大于 255。

这是 MQGET 调用的输出字段。对于 MQPUT 和 MQPUT1 调用，将忽略此参数。此字段的初始值为 0。

CodedCharSetId (MQLONG)

此字段指定消息体中字符数据的字符集标识。

注: MQMD 中的字符数据以及作为调用参数的其他 MQ 数据结构必须位于队列管理器的字符集中。此属性由队列管理器的 *CodedCharSetId* 属性定义; 请参阅第 701 页的『队列管理器的属性』以获取此属性的详细信息。

如果在选项中使用 MQGMO_CONVERT 调用 MQGET 时将此字段设置为 MQCCSI_Q_MGR, 那么客户机和服务器应用程序之间的行为有所不同。对于服务器应用程序, 用于字符转换的代码页是队列管理器的 *CodedCharSetId*; 对于客户机应用程序, 用于字符转换的代码页是当前语言环境代码页。

对于客户机应用程序, 将根据客户机的语言环境而不是队列管理器上的语言环境来填充 MQCCSI_Q_MGR。该规则的例外情况是将消息放入 IMS 网桥队列; 在 MQMD 的 *CodedCharSetId* 字段中返回的内容是队列管理器的 CCSID。

不得使用以下特殊值:

MQCCSI_APPL

当使用带有 MQGMO_CONVERT 选项的 MQGET 调用接收到消息时, 这会导致 MQMD 的 *CodedCharSetId* 字段中的值不正确, 并导致返回码 MQRC_SOURCE_CCSD_ERROR (或 MQRC_FORMAT_ERROR for z/OS)。

您可以使用以下特殊值:

MQCCSI_Q_MGR

消息中的字符数据位于队列管理器的字符集中。

在 MQPUT 和 MQPUT1 调用上, 队列管理器会将随消息一起发送的 MQMD 中的此值更改为队列管理器的真实字符集标识。因此, MQGET 调用从不返回值 MQCCSI_Q_MGR。

MQCCSI_DEFAULT

String 字段中数据的 *CodedCharSetId* 由 MQCFH 结构之前的头结构中的 *CodedCharSetId* 字段定义, 或者由 MQMD 中的 *CodedCharSetId* 字段 (如果 MQCFH 位于消息开头) 定义。

MQCCSI_INHERIT

消息中的字符数据与此结构的字符集相同; 这是队列管理器的字符集。(仅对于 MQMD, MQCCSI_INHERIT 具有与 MQCCSI_Q_MGR 相同的含义)。

队列管理器将随消息一起发送的 MQMD 中的此值更改为 MQMD 的实际字符集标识。如果未发生错误, 那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER, 请不要使用 MQCCSI_INHERIT。

MQCCSI_EMBEDDED

消息中的字符数据位于包含在消息数据本身中的标识的字符集中。可以有任意数量的字符集标识嵌入到消息数据中, 应用于数据的不同部分。此值必须用于包含混合字符集中的数据的 PCF 消息 (格式为 MQFMT_ADMIN, MQFMT_EVENT 或 MQFMT_PCF)。PCF 消息中包含的每个 MQCFST, MQCFSL 和 MQCFSF 结构都必须指定显式字符集标识, 而不是 MQCCSI_DEFAULT。

如果格式为 MQFMT_EMBEDDED_PCF 的消息要包含混合字符集中的数据, 请不要使用 MQCCSI_EMBEDDED。而是在 MQEPH 结构的标志字段中设置 MQEPH_CCSD_EMBEDDED。这相当于在前面的结构中设置 MQCCSI_EMBEDDED。然后, PCF 消息中包含的每个 MQCFST, MQCFSL 和 MQCFSF 结构都必须指定显式字符集标识, 而不是 MQCCSI_DEFAULT。有关 MQEPH 结构的更多信息, 请参阅第 319 页的『MQEPH-嵌入式 PCF 头』。

仅在 MQPUT 和 MQPUT1 调用上指定此值。如果在 MQGET 调用上指定了此参数, 那么将阻止转换消息。

在 MQPUT 和 MQPUT1 调用上, 队列管理器会更改随上述消息一起发送的 MQMD 中的 MQCCSI_Q_MGR 和 MQCCSI_INHERIT 值, 但不会更改 MQPUT 或 MQPUT1 调用上指定的 MQMD。不会对指定的值执行其他检查。

检索消息的应用程序必须将此字段与应用程序期望的值进行比较; 如果值不同, 那么应用程序可能需要转换消息中的字符数据。

如果在 MQGET 调用上指定 MQGMO_CONVERT 选项, 那么此字段是输入/输出字段。应用程序指定的值是要将消息数据转换为的编码字符集标识 (如果需要)。如果转换成功或不必要, 那么值保持不变 (除了将值 MQCCSI_Q_MGR 或 MQCCSI_INHERIT 转换为实际值)。如果转换失败, 那么 MQGET 调用后的值表示返回到应用程序的未转换消息的编码字符集标识。

否则，这是 MQGET 调用的输出字段，也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的初始值为 MQCCSI_Q_MGR。

CorrelId (MQBYTE24)

CorrelId 字段是消息头中可用于标识特定消息或消息组的属性。

这是一个字节字符串，应用程序可以使用该字节字符串将一条消息与另一条消息关联，或者将该消息与应用程序正在执行的其他工作关联。相关标识是消息的永久属性，并且在队列管理器重新启动时持久存在。由于相关标识是字节字符串而不是字符串，因此当消息从一个队列管理器流向另一个队列管理器时，不会在字符集之间转换相关标识。

对于 MQPUT 和 MQPUT1 调用，应用程序可以指定任何值。队列管理器将此值与消息一起传输，并将其传递给发出消息获取请求的应用程序。

如果应用程序指定 MQPMO_NEW_CORREL_ID，那么队列管理器将生成随消息一起发送的唯一相关标识，并在 MQPUT 或 MQPUT1 调用的输出时返回到发送应用程序。

队列管理器生成的相关标识由 3 字节产品标识 (ASCII 或 EBCDIC 中的 AMQ 或 CSQ) 组成，后跟一个保留字节以及唯一字符串的特定于产品的实现。在 WebSphere MQ 中，此特定于产品的实现字符串包含队列管理器名称的前 12 个字符以及从系统时钟派生的值。因此，所有可以相互通信的队列管理器都必须具有在前 12 个字符中不同的名称，以确保消息标识是唯一的。生成唯一字符串的能力还取决于不向后更改系统时钟。要消除队列管理器生成的消息标识与应用程序生成的消息标识重复的可能性，应用程序必须避免生成具有 ASCII 或 EBCDIC (X'41' 到 X'49' 和 X'C1' 到 X'C9') 范围内的初始字符的标识。但是，不会阻止应用程序生成这些范围中具有初始字符的标识。

此生成的相关标识将与消息一起保留 (如果保留)，并在将消息作为发布内容发送给在 MQSUB 调用上传递的 MQSD 中的 SubCorrelId 字段中指定 MQCI_NONE 的订户时用作相关标识。有关保留发布的更多信息，请参阅 [MQPMO 选项](#)。

当队列管理器或消息通道代理生成报告消息时，它会以原始消息的 *Report* 字段 (MQRO_COPY_MSG_ID_TO_CORREL_ID 或 MQRO_PASS_CORREL_ID) 指定的方式设置 *CorrelId* 字段。生成报告消息的应用程序也必须执行此操作。

对于 MQGET 调用，*CorrelId* 是可用于选择要从队列中检索的特定消息的五个字段之一。请参阅 *MsgId* 字段的描述，以获取有关如何指定此字段的值的详细信息。

指定 MQCI_NONE 作为相关标识与不指定 MQMO_MATCH_CORREL_ID 具有相同的效果，即任何相关标识都将匹配。

如果在 MQGET 调用的 *GetMsgOpts* 参数中指定了 MQGMO_MSG_UNDER_CURSOR 选项，那么将忽略此字段。

从 MQGET 调用返回时，*CorrelId* 字段将设置为返回的消息的相关标识 (如果有)。

可以使用以下特殊值：

MQCI_NONE

未指定相关标识。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQCI_NONE_ARRAY；此值与 MQCI_NONE 相同，但是字符数组而不是字符串。

MQCI_NEW_SESSION

消息是新会话的开始。

此值被 CICS 网桥识别为指示新会话的启动，即新消息序列的启动。

对于 C 编程语言，还定义了常量 MQCI_NEW_SESSION_ARRAY；此值与 MQCI_NEW_SESSION 相同，但是字符数组而不是字符串。

对于 MQGET 调用，这是输入/输出字段。对于 MQPUT 和 MQPUT1 调用，如果未指定 MQPMO_NEW_CORREL_ID，那么这是输入字段，如果指定 MQPMO_NEW_CORREL_ID，那么这是输出字段。此字段的长度由 MQ_CORREL_ID_LENGTH 给出。此字段的初始值为 MQCI_NONE。

注：

不能在层次结构中传递发布内容的相关标识。该字段供队列管理器使用。

Encoding (MQLONG)

这指定消息中数字数据的数字编码; 它不适用于 MQMD 结构本身中的数字数据。数字编码定义用于二进制整数, 压缩十进制整数和浮点数的表示。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。队列管理器不会检查该字段是否有效。定义了以下特殊值:

MQENC_NATIVE

编码是运行应用程序的编程语言和机器的缺省值。

注: 此常量的值取决于编程语言和环境。因此, 必须使用适合于应用程序将在其中运行的环境的头, 宏, COPY 或 INCLUDE 文件来编译应用程序。

放置消息的应用程序通常指定 MQENC_NATIVE。检索消息的应用程序必须将此字段与值 MQENC_NATIVE 进行比较; 如果值不同, 那么应用程序可能需要转换消息中的数字数据。使用 MQGMO_CONVERT 选项来请求队列管理器在处理 MQGET 调用的过程中转换消息。有关如何构造 *Encoding* 字段的详细信息, 请参阅第 785 页的『机器编码』。

如果在 MQGET 调用上指定 MQGMO_CONVERT 选项, 那么此字段是输入/输出字段。应用程序指定的值是要将消息数据转换为的编码(如果需要)。如果转换成功或不必要, 那么值保持不变。如果转换失败, 那么 MQGET 调用后的值表示返回到应用程序的未转换消息的编码。

在其他情况下, 这是 MQGET 调用的输出字段, 也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的初始值为 MQENC_NATIVE。

Expiry (MQLONG)

这是由放置消息的应用程序设置的以十分之一秒为单位的时间段。在此时间段经过之前, 尚未从目标队列中移除的消息将变得可被废弃。

此值将递减, 以反映消息在目标队列以及任何中间传输队列(如果放置到远程队列)上所花费的时间。它还可以由消息通道代理程序进行递减, 以反映传输时间(如果这些时间很重要)。同样, 将此消息转发到另一个队列的应用程序可能会在必要时减小该值(如果它已保留此消息相当长的时间)。但是, 到期时间被视为近似, 并且无需减小值以反映较小的时间间隔。

当应用程序使用 MQGET 调用检索消息时, *Expiry* 字段表示仍保留的原始到期时间量。

在经过消息的到期时间之后, 该消息将有资格被队列管理器废弃。当发生浏览或非浏览 MQGET 调用时, 如果消息尚未到期, 那么将废弃该消息。例如, 非浏览 MQGET 调用, 其中 MQGMO 中的 *MatchOptions* 字段设置为从 FIFO 有序队列读取 MQMO_NONE 将废弃所有到期消息, 直到第一个未到期消息为止。使用优先级排序队列时, 同一调用将废弃在第一条未到期消息之前到达队列的优先级较高的到期消息和优先级相同的消息。

不会将已到期的消息返回到应用程序(通过浏览或非浏览 MQGET 调用), 因此成功 MQGET 调用后消息描述符的 *Expiry* 字段中的值大于零或特殊值 MQEI_UNLIMITED。

如果将消息放在远程队列上, 那么在消息到达目标队列之前, 消息在中间传输队列上时可能会到期(并被废弃)。

如果消息指定了其中一个 MQRO_EXPIRATION_* 报告选项, 那么将在废弃到期消息时生成报告。如果未指定任何这些选项, 那么不会生成此类报告; 假定此消息在此时间段后不再相关(可能是因为稍后的消息已取代此消息)。

对于放在同步点内的消息, 到期时间间隔从放入消息时开始, 而不是从落实同步点时开始。在落实同步点之前, 可以通过到期时间间隔。在这种情况下, 将在落实操作之后的某个时间废弃该消息, 并且不会将该消息返回到应用程序以响应 MQGET 操作。

根据到期时间废弃消息的任何其他程序也必须发送相应的报告消息(如果已请求)。

注:

1. 如果消息的 *Expiry* 时间为零或大于 999 999 999, 那么 MQPUT 或 MQPUT1 调用将失败, 原因码为 MQRC_EXPIRY_ERROR; 在这种情况下不会生成报告消息。

2. 因为具有已经过的到期时间的消息可能要到以后才会被废弃，所以队列上可能存在已超过其到期时间的消息，因此不适合检索这些消息。但是，出于所有目的 (包括深度触发)，这些消息将计入队列中的消息数。
3. 如果请求，将在废弃消息时生成到期报告，而不是在其符合废弃条件时生成到期报告。
4. 废弃已到期的消息并在请求时生成到期报告从来不是应用程序工作单元的一部分，即使消息已调度为由于在工作单元内运行的 MQGET 调用而废弃。
5. 如果工作单元内的 MQGET 调用检索到接近到期的消息，并且随后回退了该工作单元，那么该消息可能有资格在再次检索之前被废弃。
6. 如果使用 MQGMO_LOCK 的 MQGET 调用锁定了接近到期的消息，那么在使用 MQGMO_MSG_UNDER_CURSOR 的 MQGET 调用检索该消息之前，该消息可能会被废弃；如果发生此情况，那么会在此后续 MQGET 调用上返回原因码 MQRC_NO_MSG_UNDER_CURSOR。
7. 检索到到期时间大于零的请求消息时，应用程序可以在发送应答消息时执行下列其中一项操作：
 - 将剩余到期时间从请求消息复制到应答消息。
 - 将应答消息中的到期时间设置为大于零的显式值。
 - 将应答消息中的到期时间设置为 MQEI_UNLIMITED。

要执行的操作取决于应用程序的设计。但是，将消息放入死信 (undelivered-message) 队列的缺省操作必须是保留消息的剩余到期时间，并继续将其递减。

8. 触发器消息始终使用 MQEI_UNLIMITED 生成。
9. *Format* 名称为 MQFMT_XMIT_Q_HEADER 的消息 (通常在传输队列上) 在 MQXQH 中具有第二个消息描述符。因此，它有两个关联的 *Expiry* 字段。在这种情况下，应注意以下补充要点：
 - 当应用程序将消息放在远程队列上时，队列管理器会将消息初始放在本地传输队列上，并以 MQXQH 结构作为应用程序消息数据的前缀。队列管理器将两个 *Expiry* 字段的值设置为与应用程序指定的值相同。

如果应用程序将消息直接放在本地传输队列上，那么消息数据必须已以 MQXQH 结构开头，并且格式名称必须为 MQFMT_XMIT_Q_HEADER。在这种情况下，应用程序无需将这两个 *Expiry* 字段的值设置为相同。(队列管理器会检查 MQXQH 中的 *Expiry* 字段是否包含有效值，以及消息数据的长度是否足以包含该值)。对于可以直接写入传输队列的应用程序，应用程序必须使用嵌入式消息描述符创建传输队列头。但是，如果写入传输队列的消息描述符中的到期值与嵌入式消息描述符中的值不一致，那么将发生到期错误拒绝。

 - 从队列 (无论是正常队列还是传输队列) 检索 *Format* 名称为 MQFMT_XMIT_Q_HEADER 的消息时，队列管理器会将这两个 *Expiry* 字段与在队列中等待所花费的时间一起递减。如果消息数据的长度不足以在 MQXQH 中包含 *Expiry* 字段，那么不会发生任何错误。
 - 队列管理器使用单独的消息描述符 (即，MQXQH 结构中嵌入的消息描述符中的字段) 中的 *Expiry* 字段来测试消息是否适合废弃。
 - 如果两个 *Expiry* 字段的初始值不同，那么检索消息时单独的消息描述符中的 *Expiry* 时间可能大于零 (因此消息不适合废弃)，而根据 MQXQH 中的 *Expiry* 字段的时间已过去。在这种情况下，MQXQH 中的 *Expiry* 字段设置为零。
10. 除非在 MQIIH 的 "标志" 字段中设置 MQIIH_PASS_EXPIRATION，否则从 IMS 网桥返回的应答消息的到期时间不受限制。请参阅 [标志](#) 以获取更多信息。

可识别以下特殊值：

MQEI_UNLIMITED

消息具有无限的到期时间。

这是 MQGET 调用的输出字段，也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的初始值为 MQEI_UNLIMITED。

Feedback (MQLONG)

"反馈" 字段与 MQMT_REPORT 类型的消息配合使用，以指示报告的性质，并且仅对该类型的消息有意义。

该字段可以包含其中一个 MQFB_* 值或其中一个 MQRC_* 值。反馈代码分组如下：

MQFB_NONE

未提供反馈。

MQFB_SYSTEM_FIRST

系统生成的反馈的最小值。

MQFB_SYSTEM_LAST

系统生成的反馈的最高值。

系统生成的反馈代码 MQFB_SYSTEM_FIRST 到 MQFB_SYSTEM_LAST 的范围包括本主题中列出的常规反馈代码 (MQFB_*), 以及无法将消息放入目标队列时可能发生的原因码 (MQRC_*)。

MQFB_APPL_FIRST

应用程序生成的反馈的最低值。

MQFB_APPL_LAST

应用程序生成的反馈的最高值。

生成报告消息的应用程序不得使用系统范围内的反馈代码 (MQFB_QUIT 除外), 除非它们想要模拟队列管理器或消息通道代理程序生成的报告消息。

在 MQPUT 或 MQPUT1 调用上, 指定的值必须是 MQFB_NONE, 或者在系统范围或应用程序范围内。无论 *MsgType* 的值如何, 都将选中此项。

常规反馈代码:**MQFB_COA**

确认到达目标队列 (请参阅 MQRO_COA)。

MQFB_COD

确认传递到接收应用程序 (请参阅 MQRO_COD)。

MQFB_EXPIRATION

由于未在到期时间之前将消息从目标队列中除去, 因此已废弃该消息。

MQFB_PAN

肯定操作通知 (请参阅 MQRO_PAN)。

MQFB_NAN

负面操作通知 (请参阅 MQRO_NAN)。

MQFB_QUIT

结束应用程序。

这可以由工作负载调度程序用于控制正在运行的应用程序的实例数。将带有此反馈代码的 MQMT_REPORT 消息发送到应用程序的实例向该实例指示它应该停止处理。但是, 遵守此约定是应用程序的事; 队列管理器不会强制执行此约定。

通道反馈代码:**MQFB_CHANNEL_COMPLETED**

通道正常结束。

MQFB_CHANNEL_FAIL

通道异常结束并进入 STOPPED 状态。

MQFB_CHANNEL_FAIL_RETRY

通道异常结束并进入 RETRY 状态。

IMS-bridge 反馈代码

当接收到意外的 IMS-OTMA 检测代码时, 将使用这些代码。检测代码或在检测代码为 0x1A 时与该检测代码关联的原因码在反馈中指示。

1. 对于范围在 MQFB_IMS_FIRST (300) 到 MQFB_IMS_LAST (399) 之间的反馈代码, 接收到非 0x1A 的检测代码。检测代码由表达式 (*Feedback* - MQFB_IMS_FIRST+1) 提供
2. 对于范围为 MQFB_IMS_NACK_1A_REASON_FIRST (600) 到 MQFB_IMS_NACK_1A_REASON_LAST (855) 的反馈代码, 接收到检测代码 0x1A。与检测代码关联的原因码由表达式 (*Feedback* - MQFB_IMS_NACK_1A_REASON_FIRST) 提供

Open Transaction Manager Access Guide and Reference 中描述了 IMS-OTMA 检测代码和相应原因码的含义。

IMS 网桥可以生成以下反馈代码:

MQFB_DATA_LENGTH_ZERO

消息的应用程序数据中的段长度为零。

MQFB_DATA_LENGTH_负数

消息的应用程序数据中的段长度为负数。

MQFB_DATA_LENGTH_TOO_BIG

段长度在消息的应用程序数据中过大。

MQFB_BUFFER_OVERFLOW

其中一个长度字段的值将导致数据溢出消息缓冲区。

MQFB_LENGTH_OFF_BY_ONE

其中一个长度字段的值 1 字节太短。

MQFB_IIH_ERROR

MQMD 中的 *Format* 字段指定 MQFMT_IMS，但消息未以有效的 MQIIH 结构开头。

MQFB_NOT_AUTHORIZED_FOR_IM

消息描述符 MQMD 中包含的用户标识或 MQIIH 结构中 *Authenticator* 字段中包含的密码未能通过 IMS 网桥执行的验证。因此，消息未传递到 IMS。

MQFB_IMS_ERROR

IMS 返回了意外错误。请参阅 IMS 网桥所在系统上的 WebSphere MQ 错误日志，以获取有关该错误的更多信息。

MQFB_IMS_FIRST

当 IMS-OTMA 检测代码不是 0x1A 时，IMS 生成的反馈代码在 MQFB_IMS_FIRST (300) 到 MQFB_IMS_LAST (399) 的范围内。IMS-OTMA 检测代码本身为 *Feedback* 减去 MQFB_IMS_ERROR。

MQFB_IMS_LAST

当检测代码不是 0x1A 时，IMS 生成的反馈的最大值。

MQFB_IMS_NACK_1A_REASON_FIRST

当检测代码为 0x1A 时，IMS 生成的反馈代码在 MQFB_IMS_NACK_1A_REASON_FIRST (600) 到 MQFB_IMS_NACK_1A_REASON_LAST (855) 范围内。

MQFB_IMS_NACK_1A_REASON_LAST

检测代码为 0x1A 时 IMS 生成的反馈的最大值

CICS-bridge 反馈代码: CICS 网桥可以生成以下反馈代码:

MQFB_CICS_APPL_AB 已结束

消息中指定的应用程序异常结束。此反馈代码仅出现在 MQDLH 结构的 *Reason* 字段中。

MQFB_CICS_APPL_NOT_STARTED

消息中指定的应用程序的 EXEC CICS LINK 失败。此反馈代码仅出现在 MQDLH 结构的 *Reason* 字段中。

MQFB_CICS_BRIDGE_FAILURE

CICS 网桥异常终止，但未完成正常错误处理。

MQFB_CICS_CCSID_ERROR

字符集标识无效。

MQFB_CICS_CIH_ERROR

CICS 信息头结构缺失或无效。

MQFB_CICS_COMMAREA_ERROR

CICS COMMAREA 的长度无效。

MQFB_CICS_CORREL_ID_ERROR

相关标识无效。

MQFB_CICS_DLQ_ERROR

CICS 网桥任务无法将对此请求的应答复制到死信队列。请求已回退。

MQFB_CICS_ENCODING_ERROR

编码无效。

MQFB_CICS_INTERNAL_ERROR

CICS 网桥迂到意外错误。

此反馈代码仅出现在 MQDLH 结构的 *Reason* 字段中。

MQFB_CICS_NOT_AUTHORIZED

用户标识未授权或密码无效。

此反馈代码仅出现在 MQDLH 结构的 *Reason* 字段中。

MQFB_CICS_UOW_BACKED_OUT

由于下列原因之一，工作单元已回退：

- 在同一工作单元中处理另一个请求时检测到故障。
- 工作单元正在进行时发生 CICS 异常终止。

MQFB_CICS_UOW_ERROR

工作单元控制字段 *UOWControl* 无效。

跟踪路由消息反馈代码：**MQFB_ACTIVITY**

与 MQFMT_EMBEDDED_PCF 格式配合使用，以允许在活动报告之后选择用户数据。

MQFB_MAX_ACTIVactivities

由于涉及消息的活动数超过最大活动数限制而废弃跟踪路由消息时返回。

MQFB_NOT_FORWARD

由于要将跟踪路由消息发送至不支持跟踪路由消息的远程队列管理器而废弃该消息时返回。

MQFB_NOT_交付

由于要将跟踪路由消息放入本地队列而废弃该消息时返回。

MQFB_UNSUPPORTED_转发

由于转发参数中的值无法识别并且在被拒绝的位掩码中，因此废弃跟踪路由消息时返回。

MQFB_UNSUPPORTED_DELIVERY

由于 *delivery* 参数中的值无法识别并且在被拒绝的位掩码中，因此废弃 *trace-route* 消息时返回。

WebSphere MQ 原因码：对于异常报告消息，*Feedback* 包含 WebSphere MQ 原因码。可能的原因码包括：

MQRC_PUT_禁止

(2051, X'803') 对队列禁止 Put 调用。

MQRC_Q_FULL

(2053, X'805') 队列已包含最大消息数。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_Q_SPACE_NOT_AVAILABLE

(2056, X'808') 磁盘上没有可用于队列的空间。

MQRC_PERSISTENT_NOT_ALLOWED

(2048, X'800') 队列不支持持久消息。

MQRC_MSG_TOO_BIG_FOR_Q_MGR

(2031, X'7EF') 消息长度大于队列管理器的最大长度。

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE') 消息长度大于队列的最大长度。

有关原因码的完整列表，请参阅：

- 对于 WebSphere MQ for z/OS，请参阅 [API 原因码](#)。
- 对于所有其他平台，请参阅 [API 完成代码和原因码](#)。

这是 MQGET 调用的输出字段，也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的初始值为 MQFB_NONE。

Format (MQCHAR8)

这是消息发送方用于向接收方指示消息中数据的性质的名称。可以为名称指定队列管理器字符集中的任何字符，但必须将名称限制为以下内容：

- 大写 A 到 Z
- 数字数字 0 到 9

如果使用其他字符，那么可能无法在发送和接收队列管理器的字符集之间转换名称。

将带有空格的名称填充到字段的长度上，或使用空字符在字段结束之前终止名称；将空字符和任何后续字符视为空格。不要指定带有前导空格或嵌入空格的名称。对于 MQGET 调用，队列管理器会将用空格填充的名称返回到字段的长度。

队列管理器不检查该名称是否符合上述建议。

以大写，小写和混合大小写开头的名称具有队列管理器定义的含义；请勿将以这些字母开头的名称用于您自己的格式。队列管理器内置格式为：

MQFMT_NONE

未定义数据的性质：使用 MQGMO_CONVERT 选项从队列中检索消息时，无法转换数据。

如果在 MQGET 调用上指定 MQGMO_CONVERT，并且消息中数据的字符集或编码与 *MsgDesc* 参数中指定的字符集或编码不同，那么将返回具有以下完成代码和原因码的消息（假定没有其他错误）：

- 完成代码 MQCC_WARNING 和原因码 MQRC_FORMAT_ERROR（如果 MQFMT_NONE 数据位于消息的开头）。
- 如果 MQFMT_NONE 数据位于消息末尾（即，前面有一个或多个 MQ 头结构），那么完成代码 MQCC_OK 和原因码 MQRC_NONE。在这种情况下，会将 MQ 头结构转换为请求的字符集和编码。

对于 C 编程语言，还定义了常量 MQFMT_NONE_ARRAY；此值与 MQFMT_NONE 相同，但是字符数组而不是字符串。

MQFMT_ADMIN

该消息是可编程命令格式 (PCF) 的命令-服务器请求或应答消息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。有关使用可编程命令格式消息的更多信息，请参阅 [使用可编程命令格式](#)。

对于 C 编程语言，还定义了常量 MQFMT_ADMIN_ARRAY；此值与 MQFMT_ADMIN 相同，但是字符数组而不是字符串。

MQFMT_CICS

消息数据以 CICS 信息头 MQCIH 开头，后跟应用程序数据。应用程序数据的格式名称由 MQCIH 结构中的 *Format* 字段提供。

在 z/OS 上，在 MQGET 调用上指定 MQGMO_CONVERT 选项以转换格式为 MQFMT_CICS 的消息。

对于 C 编程语言，还定义了常量 MQFMT_CICS_ARRAY；此值与 MQFMT_CICS 相同，但是字符数组而不是字符串。

MQFMT_COMMAND_1

此消息是包含对象计数，完成代码和原因码的 MQSC 命令服务器应答消息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

对于 C 编程语言，还定义了常量 MQFMT_COMMAND_1_ARRAY；此值与 MQFMT_COMMAND_1 相同，但是字符数组而不是字符串。

MQFMT_COMMAND_2

此消息是 MQSC 命令服务器应答消息，其中包含有关所请求对象的信息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

对于 C 编程语言，还定义了常量 MQFMT_COMMAND_2_ARRAY；此值与 MQFMT_COMMAND_2 相同，但是字符数组而不是字符串。

MQFMT_DEAD_LETTER_HEADER

消息数据以死信头 MQDLH 开头。原始消息中的数据紧跟 MQDLH 结构。原始消息数据的格式名称由 MQDLH 结构中的 *Format* 字段提供；请参阅第 307 页的『MQDLH-死信头』以获取此结构的详细信息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

对于 *Format* 为 MQFMT_DEAD_LETTER_HEADER 的消息，不会生成 COA 和 COD 报告。

对于 C 编程语言，还定义了常量 MQFMT_DEAD_LETTER_HEADER_ARRAY；此值与 MQFMT_DEAD_LETTER_HEADER 相同，但是字符数组而不是字符串。

MQFMT_DIST_HEADER

消息数据以 distribution-list 头 MQDH 开头；这包括 MQOR 和 MQPMR 记录的数组。可以在 distribution-list 头后面添加其他数据。附加数据 (如果有) 的格式由 MQDH 结构中的 *Format* 字段提供；请参阅第 302 页的『MQDH - 分发头』以获取此结构的详细信息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换格式为 MQFMT_DIST_HEADER 的消息。

此格式在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

对于 C 编程语言，还定义了常量 MQFMT_DIST_HEADER_ARRAY；此值与 MQFMT_DIST_HEADER 相同，但是字符数组而不是字符串。

MQFMT_EMBEDDED_PCF

跟踪路由消息的格式，前提是 PCF 命令值设置为 MQCMD_TRACE_ROUTE。使用此格式允许随跟踪路由消息一起发送用户数据，前提是他们的应用程序可以处理先前的 PCF 参数。

PCF 头 **必须** 是第一个头，否则消息将不会被视为跟踪路由消息。这意味着消息不能在组中，并且跟踪路由消息不能分段。如果在组中发送跟踪路由消息，那么将拒绝此消息，原因码为 MQRC_MSG_NOT_ALLOWED_IN_GROUP。

请注意，MQFMT_ADMIN 也可以用于跟踪路由消息的格式，但在这种情况下，不能将任何用户数据与跟踪路由消息一起发送。

MQFMT_EVENT

该消息是报告发生的事件的 MQ 事件消息。事件消息具有与可编程命令相同的结构；请参阅 [PCF 命令消息](#) 以获取有关此结构的更多信息，并参阅 [事件监视](#) 以获取有关事件的信息。

如果在 MQGET 调用上指定了 Mqgmo_convert 选项，那么可以在所有环境中转换 Version-1 事件消息。Version-2 事件消息只能在 z/OS 上进行转换。

对于 C 编程语言，还定义了常量 MQFMT_EVENT_ARRAY；此值与 MQFMT_EVENT 相同，但是字符数组而不是字符串。

MQFMT_IMS

消息数据以 IMS 信息头 MQIIH 开头，后跟应用程序数据。应用程序数据的格式名称由 MQIIH 结构中的 *Format* 字段提供。

有关将 MQGET 与 MQGMO_CONVERT 配合使用时如何处理 MQIIH 结构的详细信息，请参阅第 352 页的『Format (MQCHAR8)』和 第 353 页的『ReplyTo 格式 (MQCHAR8)』。

对于 C 编程语言，还定义了常量 MQFMT_IMS_ARRAY；此值与 MQFMT_IMS 相同，但是字符数组而不是字符串。

MQFMT_IMS_VAR_STRING

消息是 IMS 变量字符串，它是格式为 llzzccc 的字符串，其中：

ll

是一个 2 字节长度字段，用于指定 IMS 变量字符串项的总长度。此长度等于 ll (2 字节) 的长度加上 zz (2 字节) 的长度加上字符串本身的长度。ll 是由 *Encoding* 字段指定的编码中的 2 字节二进制整数。

zz

是包含对 IMS 很重要的标志的 2 字节字段。zz 是由两个 MQBYTE 字段组成的字节字符串，传输时不会从发送方更改为接收方 (即，zz 不受任何转换限制)。

ccc

是包含 11-4 个字符的变长字符串。ccc 位于 *CodedCharSetId* 字段指定的字符集中。

在 z/OS 上，消息数据可以由一系列 IMS 变量字符串对接在一起，每个字符串的格式为 11zzccc。在连续的 IMS 变量字符串之间不得跳过任何字节。这意味着，如果第一个字符串的长度为奇数，那么第二个字符串将不对齐，即，它不会在两个的倍数的边界上开始。在需要基本数据类型对齐的机器上构造此类字符串时请小心。

在 MQGET 调用上使用 MQGMO_CONVERT 选项来转换格式为 MQFMT_IMS_VAR_STRING 的消息。

对于 C 编程语言，还定义了常量 MQFMT_IMS_VAR_STRING_ARRAY；此值与 MQFMT_IMS_VAR_STRING 相同，但是字符数组而不是字符串。

MQFMT_MD_EXTENSION

消息数据以消息描述符扩展 MQMDE 开头，并可选择后跟其他数据（通常是应用程序消息数据）。

MQMDE 之后的数据的格式名称，字符集和编码由 MQMDE 中的 *Format*，*CodedCharSetId* 和 *Encoding* 字段提供。请参阅第 406 页的『MQMDE-消息描述符扩展』以获取此结构的详细信息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

对于 C 编程语言，还定义了常量 MQFMT_MD_EXTENSION_ARRAY；此值与 MQFMT_MD_EXTENSION 相同，但是字符数组而不是字符串。

MQFMT_PCF

该消息是符合可编程命令格式 (PCF) 消息的结构的用户定义的消息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。有关使用可编程命令格式消息的更多信息，请参阅 [使用可编程命令格式](#)。

对于 C 编程语言，还定义了常量 MQFMT_PCF_ARRAY；此值与 MQFMT_PCF 相同，但是字符数组而不是字符串。

MQFMT_REF_MSG_HEADER

消息数据以参考消息头 MQRMH 开头，并且可以选择后跟其他数据。数据的格式名称，字符集和编码由 MQRMH 中的 *Format*，*CodedCharSetId* 和 *Encoding* 字段提供。请参阅第 476 页的『MQRMH - 参考消息头』以获取此结构的详细信息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

此格式在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

对于 C 编程语言，还定义了常量 MQFMT_REF_MSG_HEADER_ARRAY；此值与 MQFMT_REF_MSG_HEADER 相同，但它是字符数组而不是字符串。

MQFMT_RF_HEADER

消息数据以规则和格式化头 MQRFH 开头，并且可以选择后跟其他数据。数据的格式名称，字符集和编码 (如果有) 由 MQRFH 中的 *Format*，*CodedCharSetId* 和 *Encoding* 字段提供。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

对于 C 编程语言，还定义了常量 MQFMT_RF_HEADER_ARRAY；此值与 MQFMT_RF_HEADER 相同，但是字符数组而不是字符串。

MQFMT_RF_HEADER_2

消息数据以 version-2 规则和格式化头 MQRFH2 开头，并可选择后跟其他数据。可选数据 (如果有) 的格式名称，字符集和编码由 MQRFH2 中的 *Format*，*CodedCharSetId* 和 *Encoding* 字段提供。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

对于 C 编程语言，还定义了常量 MQFMT_RF_HEADER_2_ARRAY；此值与 MQFMT_RF_HEADER_2 相同，但是字符数组而不是字符串。

MQFMT_STRING

应用程序消息数据可以是 SBCS 字符串 (单字节字符集) 或 DBCS 字符串 (双字节字符集)。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，那么可以转换此格式的消息。

对于 C 编程语言，还定义了常量 MQFMT_STRING_ARRAY；此值与 MQFMT_STRING 相同，但是字符数组而不是字符串。

MQFMT_TRIGGER

此消息是 MQTM 结构描述的触发器消息; 请参阅 第 522 页的『MQTM-触发器消息』以获取此结构的详细信息。如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项, 那么可以转换此格式的消息。

对于 C 编程语言, 还定义了常量 MQFMT_TRIGGER_ARRAY; 此值与 MQFMT_TRIGGER 相同, 但是字符数组而不是字符串。

MQFMT_WORK_INFO_HEADER

消息数据以工作信息头 MQWIIH 开头, 后跟应用程序数据。应用程序数据的格式名称由 MQWIIH 结构中的 *Format* 字段提供。

在 z/OS 上, 在 MQGET 调用上指定 MQGMO_CONVERT 选项, 以转换格式为 MQFMT_WORK_INFO_HEADER 的消息中的用户数据。但是, MQWIIH 结构本身始终以队列管理器的字符集和编码返回 (即, 无论是否指定了 MQGMO_CONVERT 选项, 都会转换 MQWIIH 结构)。

对于 C 编程语言, 还定义了常量 MQFMT_WORK_INFO_HEADER_ARRAY; 此值与 MQFMT_WORK_INFO_HEADER 相同, 但是字符数组而不是字符串。

MQFMT_XMIT_Q_HEADER

消息数据以传输队列头 MQXQH 开头。原始消息中的数据紧跟 MQXQH 结构。原始消息数据的格式名称由 MQMD 结构中的 *Format* 字段提供, 该字段是传输队列头 MQXQH 的一部分。请参阅 第 540 页的『MQXQH-传输队列头』以获取此结构的详细信息。

对于 *Format* 为 MQFMT_XMIT_Q_HEADER 的消息, 不会生成 COA 和 COD 报告。

对于 C 编程语言, 还定义了常量 MQFMT_XMIT_Q_HEADER_ARRAY; 此值与 MQFMT_XMIT_Q_HEADER 相同, 但是字符数组而不是字符串。

这是 MQGET 调用的输出字段, 也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的长度由 MQ_FORMAT_LENGTH 指定。此字段的初始值为 MQFMT_NONE。

GroupId (MQBYTE24)

这是一个字节字符串, 用于标识物理消息所属的特定消息组或逻辑消息。如果消息允许分段, 那么也会使用 *GroupId*。在所有这些情况下, *GroupId* 都具有非空值, 并且在 *MsgFlags* 字段中设置了以下一个或多个标志:

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTATION_ALLOWED

如果未设置任何这些标志, 那么 *GroupId* 具有特殊空值 MQGI_NONE。

在以下情况下, 应用程序不需要在 MQPUT 或 MQGET 调用上设置此字段:

- 在 MQPUT 调用上, 指定了 MQPMO_LOGICAL_ORDER。
- 在 MQGET 调用上, 未指定 MQMO_MATCH_GROUP_ID。

这些是针对非报告消息的消息使用这些调用的建议方法。但是, 如果应用程序需要更多控制, 或者调用是 MQPUT1, 那么应用程序必须确保将 *GroupId* 设置为适当的值。

仅当组标识唯一时, 才能正确处理消息组和段。因此, 应用程序不得生成其自己的组标识; 相反, 应用程序必须执行下列其中一项操作:

- 如果指定了 MQPMO_LOGICAL_ORDER, 那么队列管理器将自动为逻辑消息的组或段中的第一条消息生成唯一组标识, 并将该组标识用于逻辑消息的组或段中的其余消息, 因此应用程序不需要执行任何特殊操作。这是建议的过程。
- 如果未指定 MQPMO_LOGICAL_ORDER, 那么应用程序必须通过在逻辑消息的组或段中的消息的第一个 MQPUT 或 MQPUT1 调用上将 *GroupId* 设置为 MQGI_NONE, 请求队列管理器生成组标识。然后, 队列管理器在该调用的输出上返回的组标识必须用于逻辑消息的组或段中的其余消息。如果消息组包含分段消息, 那么必须将同一组标识用于该组中的所有分段和消息。

如果未指定 MQPMO_LOGICAL_ORDER，那么可以按任何顺序(例如，按反向顺序)放置逻辑消息组和段中的消息，但必须由针对其中任何消息发出的 *first* MQPUT 或 MQPUT1 调用分配组标识。

在 MQPUT 和 MQPUT1 调用的输入上，队列管理器使用队列上的物理顺序中描述的值。在 MQPUT 和 MQPUT1 调用的输出上，如果打开的对象是单个队列而不是分发列表，但如果打开的对象是分发列表，那么队列管理器会将此字段设置为随消息一起发送的值。在后一种情况下，如果应用程序需要知道生成的组标识，那么应用程序必须提供包含 *GroupId* 字段的 MQPMR 记录。

在 MQGET 调用的输入上，队列管理器使用第 340 页的表 506 中描述的值。在 MQGET 调用的输出上，队列管理器将此字段设置为检索到的消息的值。

定义了以下特殊值:

MQGI_NONE

未指定组标识。

对于字段的长度，该值为二进制零。这是用于不在组中，不在逻辑消息段中且不允许分段的消息的值。

对于 C 编程语言，还定义了常量 MQGI_NONE_ARRAY; 此值与 MQGI_NONE 相同，但是字符数组而不是字符串。

此字段的长度由 MQ_GROUP_ID_LENGTH 指定。此字段的初始值为 MQGI_NONE。如果 *Version* 小于 MQMD_VERSION_2，那么将忽略此字段。

MsgFlags (MQLONG)

MsgFlags 是用于指定消息属性或控制其处理的标志。

MsgFlags 分为以下类别:

- 分段标志
- 状态标志

分段标志: 当消息对于队列过大时，将消息放入队列的尝试通常会失败。分段是一种技术，通过此技术，队列管理器或应用程序将消息分割成称为段的小块，并将队列上的每个段作为单独的物理消息放置。检索消息的应用程序可以逐个检索段，也可以请求队列管理器将段重新组合到 MQGET 调用返回的单个消息中。后者是通过在 MQGET 调用上指定 MQGMO_COMPLETE_MSG 选项并提供足以容纳完整消息的缓冲区来实现的。(请参阅第 324 页的『MQGMO-Get-消息选项』以获取 MQGMO_COMPLETE_MSG 选项的详细信息。)可以在发送队列管理器，中间队列管理器或目标队列管理器上对消息进行分段。

您可以指定下列其中一项以控制消息的分段:

MQMF_SEGMENTATION_ALLOWED

此选项可防止队列管理器将消息分解成段。如果为已经是分段的消息指定了此选项，那么此选项将防止分段分解为较小的分段。

此标志的值为二进制零。这是缺省值。

MQMF_SEGMENTATION_ALLOWED

此选项允许队列管理器将消息分为多个段。如果为已经是段的消息指定了此选项，那么此选项允许将段细分为更小的段。可以在未设置 MQMF_SEGMENT 或 MQMF_LAST_SEGMENT 的情况下设置 MQMF_SEGMENTATION_ALLOWED。

- 在 z/OS 上，队列管理器不支持消息分段。如果消息对于队列太大，那么 MQPUT 或 MQPUT1 调用将失败，原因码为 MQRC_MSG_TOO_BIG_FOR_Q。但是，仍可以指定 MQMF_SEGMENTATION_ALLOWED 选项，并允许在远程队列管理器上对消息进行分段。

当队列管理器对消息进行分段时，队列管理器会在随每个分段一起发送的 MQMD 副本中打开 MQMF_SEGMENT 标志，但不会更改应用程序在 MQPUT 或 MQPUT1 调用上提供的 MQMD 中这些标志的设置。对于逻辑消息中的最后一个段，队列管理器还会在随该段一起发送的 MQMD 中打开 MQMF_LAST_SEGMENT 标志。

注: 在使用 MQMF_SEGMENTATION_ALLOWED 但没有 MQPMO_LOGICAL_ORDER 的情况下放置消息时请小心。如果消息为:

- 不是客户细分，

- 不在一个组中,
- 未转发,

在每个 MQPUT 或 MQPUT1 调用之前, 应用程序必须将 *GroupId* 字段重置为 MQGI_NONE, 以便队列管理器可以为每条消息生成唯一的组标识。如果未执行此操作, 那么不相关的消息可能具有相同的组标识, 这可能导致后续不正确的处理。请参阅 *GroupId* 字段和 MQPMO_LOGICAL_ORDER 选项的描述, 以获取有关何时重置 *GroupId* 字段的更多信息。

队列管理器根据需要将消息分割成段, 以便段 (以及任何必需的头数据) 适合队列。但是, 队列管理器生成的段的大小有一个下限, 只有从消息创建的最后一个段才能小于此限制 (应用程序生成的段的大小下限为一个字节)。队列管理器生成的段的长度可能不相等。队列管理器按如下所示处理消息:

- 用户定义的格式在 16 个字节的倍数的边界上拆分; 队列管理器不会生成小于 16 个字节的段 (最后一个段除外)。
- 除 MQFMT_STRING 以外的内置格式在适合于现有数据的性质的点处进行拆分。但是, 队列管理器从不拆分 WebSphere MQ 头结构中间的消息。这意味着包含单个 MQ 头结构的段不能由队列管理器进一步拆分, 因此该消息的最小可能段大小大于 16 字节。

队列管理器生成的第二个或更高版本的段以下列其中一项开始:

- MQ 头结构
- 应用程序消息数据的开始
- 通过应用程序消息数据的部分方法
- MQFMT_STRING 是拆分的, 而不考虑存在的数据的性质 (SBCS, DBCS 或混合 SBCS/DBCS)。当字符串为 DBCS 或混合 SBCS/DBCS 时, 这可能导致段无法从一个字符集转换为另一个字符集。队列管理器从不将 MQFMT_STRING 消息分割为小于 16 字节的段 (最后一个段除外)。
- 队列管理器在每个段的 MQMD 中设置 *Format*, *CodedCharSetId* 和 *Encoding* 字段, 以正确描述段的 *start* 处存在的数据; 格式名称是内置格式的名称或用户定义的格式的名称。
- 将修改 *Offset* 大于零的段的 MQMD 中的 *Report* 字段。对于每种报告类型, 如果报告选项为 MQRO_*_WITH_DATA, 但段不能包含前 100 个字节的任何用户数据 (即, 可能存在的任何 WebSphere MQ 头结构之后的数据), 那么报告选项将更改为 MQRO_*。

队列管理器遵循上述规则, 但以其他方式无法预测地拆分消息; 请勿对拆分消息的位置进行假设。

对于持久消息, 队列管理器只能在工作单元中执行分段:

- 如果 MQPUT 或 MQPUT1 调用在用户定义的工作单元中运行, 那么将使用该工作单元。如果在分段过程中调用失败, 那么队列管理器将除去由于调用失败而放置在队列上的任何段。但是, 此故障不会阻止成功落实工作单元。
- 如果调用在用户定义的工作单元外部运行, 并且不存在用户定义的工作单元, 那么队列管理器仅在调用期间创建工作单元。如果调用成功, 那么队列管理器将自动落实工作单元。如果调用失败, 那么队列管理器将回退工作单元。
- 如果调用在用户定义的工作单元外部运行, 但存在用户定义的工作单元, 那么队列管理器无法执行分段。如果消息不需要分段, 那么调用仍可成功。但是, 如果消息需要分段, 那么调用将失败, 原因码为 MQRC_UOW_NOT_AVAILABLE。

对于非持久消息, 队列管理器不需要工作单元即可执行分段。

转换可能分段的消息中的数据时, 请特别小心:

- 如果接收应用程序在 MQGET 调用上转换数据, 并指定 MQGMO_COMPLETE_MSG 选项, 那么将向数据转换出口传递完整消息以供该出口转换, 并且该消息已分段的事实对于该出口很明显。
- 如果接收应用程序一次检索一个段, 那么将调用数据转换出口以一次转换一个段。因此, 出口必须独立于任何其他段中的数据来转换段中的数据。

如果消息中数据的性质使 16 字节边界上的数据的任意分段可能导致无法由出口转换的段, 或者格式为 MQFMT_STRING 且字符集为 DBCS 或混合 SBCS/DBCS, 那么发送应用程序必须创建并放置这些段, 指定 MQMF_SEG 域禁止以禁止进一步分段。这样, 发送应用程序可以确保每个分段包含足够的信息, 以允许数据转换出口成功转换分段。

- 如果为发送消息通道代理程序 (MCA) 指定了发送方转换, 那么 MCA 仅转换不是逻辑消息段的消息; MCA 从不尝试转换作为段的消息。

此标志是 MQPUT 和 MQPUT1 调用上的输入标志, 以及 MQGET 调用上的输出标志。在后一个调用中, 队列管理器还会将该标志的值回传到 MQGMO 中的 *Segmentation* 字段。

此标志的初始值为 MQMF_SEGMENTATION_ALLOWED。

状态标志: 这些标志指示物理消息是属于消息组, 是逻辑消息的段, 两者都是, 还是两者都不是。可以在 MQPUT 或 MQPUT1 调用上指定以下一项或多项, 也可以由 MQGET 调用返回:

MQMF_MSG_IN_GROUP

消息是组的成员。

MQMF_LAST_MSG_IN_GROUP

消息是组中的最后一条逻辑消息。

如果设置了此标志, 那么队列管理器将在随消息一起发送的 MQMD 副本中打开 MQMF_MSG_IN_GROUP, 但不会更改应用程序在 MQPUT 或 MQPUT1 调用上提供的 MQMD 中这些标志的设置。

仅由一条逻辑消息组成组是有效的。如果是这种情况, 那么将设置 MQMF_LAST_MSG_IN_GROUP, 但 *MsgSeqNumber* 字段具有值 1。

MQMF_SEGMENT

消息是逻辑消息的段。

当指定 MQMF_SEGMENT 而不指定 MQMF_LAST_SEGMENT 时, 段中应用程序消息数据的长度 (不包括可能存在的任何 WebSphere MQ 头结构的长度) 必须至少为 1。如果长度为零, 那么 MQPUT 或 MQPUT1 调用将失败, 原因码为 MQRC_SEGMENT_LENGTH_ZERO。

在 z/OS 上, 如果消息正在放入索引类型为 MQIT_GROUP_ID 的队列中, 那么不支持此选项。

MQMF_LAST_SEGMENT

消息是逻辑消息的最后一个段。

如果设置了此标志, 那么队列管理器将在随消息一起发送的 MQMD 副本中打开 MQMF_SEGMENT, 但不会更改应用程序在 MQPUT 或 MQPUT1 调用上提供的 MQMD 中这些标志的设置。

逻辑消息只能由一个段组成。如果是这样, 那么将设置 MQMF_LAST_SEGMENT, 但 *Offset* 字段的值为零。

当指定 MQMF_LAST_SEGMENT 时, 段中应用程序消息数据的长度 (不包括可能存在的任何头结构的长度) 可以为零。

在 z/OS 上, 如果消息正在放入索引类型为 MQIT_GROUP_ID 的队列中, 那么不支持此选项。

应用程序必须确保在放置消息时正确设置这些标志。如果指定了 MQPMO_LOGICAL_ORDER, 或者在队列句柄的先前 MQPUT 调用上指定了 MQPMO_LOGICAL_ORDER, 那么标志的设置必须与队列管理器为队列句柄保留的组和段信息一致。当指定 MQPMO_LOGICAL_ORDER 时, 以下条件适用于队列句柄的连续 MQPUT 调用:

- 如果没有当前组或逻辑消息, 那么所有这些标志 (及其组合) 都有效。
- 指定 MQMF_MSG_IN_GROUP 后, 它必须保持开启状态, 直到指定 MQMF_LAST_MSG_IN_GROUP 为止。如果不满足此条件, 那么调用将失败, 原因码为 MQRC_INCOMPLETE_GROUP。
- 指定 MQMF_SEGMENT 后, 它必须保持开启状态, 直到指定 MQMF_LAST_SEGMENT 为止。如果不满足此条件, 那么调用将失败, 原因码为 MQRC_INCOMPLETE_MSG。
- 在未指定 MQMF_MSG_IN_GROUP 的情况下指定 MQMF_SEGMENT 后, MQMF_MSG_IN_GROUP 必须保持关闭状态, 直到指定 MQMF_LAST_SEGMENT 为止。如果不满足此条件, 那么调用将失败, 原因码为 MQRC_INCOMPLETE_MSG。

队列上的物理顺序 显示标志的有效组合以及用于各种字段的值。

这些标志是 MQPUT 和 MQPUT1 调用上的输入标志, 以及 MQGET 调用上的输出标志。在后一个调用中, 队列管理器还会将标志的值回传到 MQGMO 中的 *GroupStatus* 和 *SegmentStatus* 字段。

不能将分组或分段消息与发布/预订配合使用。

缺省标志: 可以指定以下内容以指示消息具有缺省属性:

MQMF_NONE

无消息标志 (缺省消息属性)。

这将禁止分段, 并指示消息不在组中并且不是逻辑消息的段。定义 MQMF_NONE 以帮助程序文档。不打算将此标志与任何其他标志一起使用, 但由于其值为零, 因此无法检测到此类使用。

MsgFlags 字段被划分为子字段; 有关详细信息, 请参阅第 788 页的『报告选项和消息标志』。

此字段的初始值为 MQMF_NONE。如果 *Version* 小于 MQMD_VERSION_2, 那么将忽略此字段。

MsgId (MQBYTE24)

这是用于区分一条消息和另一条消息的字节字符串。通常, 任何两条消息都不应具有相同的消息标识, 尽管队列管理器不允许这样做。消息标识是消息的永久属性, 并且在队列管理器重新启动时持久存在。由于消息标识是字节字符串而不是字符串, 因此当消息从一个队列管理器流向另一个队列管理器时, 不会在字符集之间转换消息标识。

对于 MQPUT 和 MQPUT1 调用, 如果应用程序指定了 MQMI_NONE 或 MQPMO_NEW_MSG_ID, 那么队列管理器将生成唯一消息标识² 当消息被放入时, 将其放入随消息一起发送的消息描述符中。队列管理器还会在属于发送应用程序的消息描述符中返回此消息标识。应用程序可以使用此值来记录有关特定消息的信息, 以及响应来自应用程序其他部分的查询。

如果正在将消息放入主题中, 那么队列管理器将根据发布的每条消息的需要生成唯一消息标识。如果应用程序指定了 MQPMO_NEW_MSG_ID, 那么队列管理器将生成唯一的消息标识以在输出时返回。如果应用程序指定 MQMI_NONE, 那么 MQMD 中 *MsgId* 字段的值在从调用返回时保持不变。

请参阅第 438 页的『MQPMO 选项 (MQLONG)』中 MQPMO_RETAIN 的描述, 以获取有关保留发布的更多详细信息。

如果将消息放入分发列表, 那么队列管理器将根据需要生成唯一消息标识, 但 MQMD 中 *MsgId* 字段的值在从调用返回时保持不变, 即使指定了 MQMI_NONE 或 MQPMO_NEW_MSG_ID 也是如此。如果应用程序需要知道队列管理器生成的消息标识, 那么应用程序必须提供包含 *MsgId* 字段的 MQPMR 记录。

发送应用程序还可以为除 MQMI_NONE 以外的消息标识指定值; 这将停止队列管理器生成唯一消息标识。正在转发消息的应用程序可以使用此消息来传播原始消息的消息标识。

队列管理器不使用此字段, 除非:

- 生成唯一值 (如果请求), 如上文所述
- 向发出消息的 *get* 请求的应用程序交付值
- 将该值复制到它生成的有关此消息的任何报告消息的 *CorrelId* 字段 (取决于 *Report* 选项)

当队列管理器或消息通道代理生成报告消息时, 它将以原始消息的 *Report* 字段 (MQRO_NEW_MSG_ID 或 MQRO_PASS_MSG_ID) 指定的方式设置 *MsgId* 字段。生成报告消息的应用程序也必须执行此操作。

对于 MQGET 调用, *MsgId* 是可用于从队列中检索特定消息的五个字段之一。通常, MQGET 调用会返回队列中的下一条消息, 但可以通过指定一个或多个五个选择标准 (以任意组合) 来获取特定消息; 这些字段为:

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

² 队列管理器生成的 *MsgId* 由 4 字节产品标识 (ASCII 或 EBCDIC 中的 AMQ - 或 CSQ -, 其中 - 表示空白字符) 组成, 后跟特定于产品的唯一字符串实现。在 WebSphere MQ 中, 这包含队列管理器名称的前 12 个字符以及从系统时钟派生的值。因此, 所有可以相互通信的队列管理器都必须具有在前 12 个字符中不同的名称, 以确保消息标识是唯一的。生成唯一字符串的能力还取决于不向后更改系统时钟。要消除队列管理器生成的消息标识与应用程序生成的消息标识重复的可能性, 应用程序必须避免生成具有 ASCII 或 EBCDIC (X'41 '到 X'49' 和 X'C1'到 X'C9') 范围内的初始字符的标识。但是, 不会阻止应用程序生成这些范围中具有初始字符的标识。

应用程序将这些字段中的一个或多个字段设置为所需的值，然后在 MQGMO 中的 *MatchOptions* 字段中设置相应的 MQMO_* 匹配选项以将这些字段用作选择标准。只有在这些字段中具有指定值的消息才是要检索的候选项。*MatchOptions* 字段 (如果未被应用程序改变) 的缺省值是同时匹配消息标识和相关标识。

在 z/OS 上，您可以使用的选择标准受用于队列的索引类型的限制。请参阅 *IndexType* 队列属性以获取更多详细信息。

通常，返回的消息是队列上满足选择条件的第一条消息。但是，如果指定了 MQGMO_BROWSE_NEXT，那么返回的消息是满足选择条件的 *next* 消息；此消息的扫描将从跟随当前光标位置的消息开始。

注：将按顺序扫描队列以查找满足选择标准的消息，因此检索时间比未指定选择标准时要慢，尤其是在找到合适的消息之前必须扫描许多消息。例外情况如下：

- *CorrelId* 在 64 位分布式平台上执行 MQGET 调用，其中 *CorrelId* 索引不需要执行真正的顺序扫描。
- 由 *IndexType* 在 z/OS 上执行的 MQGET 调用。

在这两种情况下，都提高了检索性能。

有关如何在各种情况下使用选择标准的更多信息，请参阅第 340 页的表 506。

将 MQMI_NONE 指定为消息标识与不指定 MQMO_MATCH_MSG_ID (即任何消息标识匹配) 具有相同的效果。

如果在 MQGET 调用的 *GetMsgOpts* 参数中指定了 MQGMO_MSG_UNDER_CURSOR 选项，那么将忽略此字段。

从 MQGET 调用返回时，*MsgId* 字段将设置为返回的消息的消息标识 (如果有)。

可以使用以下特殊值：

MQMI_NONE

未指定消息标识。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQMI_NONE_ARRAY；此值与 MQMI_NONE 相同，但是字符数组而不是字符串。

这是 MQGET，MQPUT 和 MQPUT1 调用的输入/输出字段。此字段的长度由 MQ_MSG_ID_LENGTH 给出。此字段的初始值为 MQMI_NONE。

MsgSeqNumber (MQLONG)

这是组中逻辑消息的序号。

序号从 1 开始，并针对组中的每条新逻辑消息增加 1，最多为 999 999 999 999。不在组中的物理消息的序号是 1。

在以下情况下，应用程序不必在 MQPUT 或 MQGET 调用上设置此字段：

- 在 MQPUT 调用上，指定了 MQPMO_LOGICAL_ORDER。
- 在 MQGET 调用上，未指定 MQMO_MATCH_MSG_SEQ_NUMBER。

这些是针对非报告消息的消息使用这些调用的建议方法。但是，如果应用程序需要更多控制，或者调用是 MQPUT1，那么应用程序必须确保将 *MsgSeqNumber* 设置为适当的值。

在 MQPUT 和 MQPUT1 调用的输入上，队列管理器使用 队列上的物理顺序中描述的值。在 MQPUT 和 MQPUT1 调用的输出中，队列管理器将此字段设置为随消息一起发送的值。

在 MQGET 调用的输入上，队列管理器使用第 340 页的表 506 中显示的值。在 MQGET 调用的输出上，队列管理器将此字段设置为检索到的消息的值。

此字段的初始值为 1。如果 *Version* 小于 MQMD_VERSION_2，那么将忽略此字段。

MsgType (MQLONG)

这指示消息的类型。消息类型分组如下：

MQMT_SYSTEM_FIRST

系统定义的消息类型的最小值。

MQMT_SYSTEM_LAST

系统定义的消息类型的最大值。

当前在系统范围内定义了以下值:

MQMT_DATAGRAM

消息是不需要应答的消息。

MQMT_REQUEST

消息是需要应答的消息。

在 *ReplyToQ* 字段中指定要向其发送应答的队列的名称。 *Report* 字段指示如何设置应答的 *MsgId* 和 *CorrelId*。

MQMT_REPLY

该消息是对先前请求消息 (MQMT_REQUEST) 的应答。 必须将消息发送到请求消息的 *ReplyToQ* 字段所指示的队列。 使用请求的 *Report* 字段来控制如何设置应答的 *MsgId* 和 *CorrelId*。

注: 队列管理器不会强制实施请求/应答关系; 这是应用程序的责任。

MQMT_REPORT

该消息正在报告某些预期或意外情况, 通常与其他消息相关 (例如, 接收到包含无效数据的请求消息)。 将消息发送到原始消息的消息描述符的 *ReplyToQ* 字段所指示的队列。 设置 *Feedback* 字段 *s* 以指示报告的性质。 使用原始消息的 *Report* 字段来控制如何设置报告消息的 *MsgId* 和 *CorrelId*。

队列管理器或消息通道代理程序生成的报告消息始终发送到 *ReplyToQ* 队列, 并如上所述设置 *Feedback* 和 *CorrelId* 字段。

还可以使用应用程序定义的值。 它们必须在以下范围内:

MQMT_APPL_FIRST

应用程序定义的消息类型的最小值。

MQMT_APPL_LAST

应用程序定义的消息类型的最大值。

对于 MQPUT 和 MQPUT1 调用, *MsgType* 值必须在系统定义的范围内或应用程序定义的范围内; 否则, 调用将失败, 原因码为 MQRC_MSG_TYPE_ERROR。

这是 MQGET 调用的输出字段, 也是 MQPUT 和 MQPUT1 调用的输入字段。 此字段的初始值为 MQMT_DATAGRAM。

Offset (MQLONG)

这是物理消息中数据从数据构成部分的逻辑消息开始的偏移量 (以字节为单位)。 此数据称为段。 偏移量在 0 到 999 999 999 范围内。 非逻辑消息段的物理消息的偏移量为零。

在以下情况下, 应用程序不需要在 MQPUT 或 MQGET 调用上设置此字段:

- 在 MQPUT 调用上, 指定了 MQPMO_LOGICAL_ORDER。
- 在 MQGET 调用上, 未指定 MQMO_MATCH_OFFSET。

这些是针对非报告消息的消息使用这些调用的建议方法。 但是, 如果应用程序不符合这些条件, 或者调用为 MQPUT1, 那么应用程序必须确保将 *Offset* 设置为相应的值。

在 MQPUT 和 MQPUT1 调用的输入上, 队列管理器使用 队列上的物理顺序 中描述的值。 在 MQPUT 和 MQPUT1 调用的输出中, 队列管理器将此字段设置为随消息一起发送的值。

对于报告逻辑消息段的报告消息, 将使用 *OriginalLength* 字段 (前提不是 MQOL_UNDEFINED) 来更新队列管理器保留的段信息中的偏移量。

在 MQGET 调用的输入上, 队列管理器使用 第 340 页的表 506 中显示的值。 在 MQGET 调用的输出上, 队列管理器将此字段设置为检索到的消息的值。

此字段的初始值为零。 如果 *Version* 小于 MQMD_VERSION_2, 那么将忽略此字段。

OriginalLength (MQLONG)

此字段仅与作为段的报告消息相关。它指定与报告消息相关的消息段的长度; 它不指定段构成部分的逻辑消息的长度, 也不指定报告消息中数据的长度。

注: 为作为段的消息生成报告消息时, 队列管理器和消息通道代理程序会将原始消息中的 *GroupId*, *MsgSeqNumber*, *Offset* 和 *MsgFlags* 字段复制到报告消息的 MQMD 中。因此, 报告消息也是一个分段。生成报告消息的应用程序必须执行相同的操作, 并正确设置 *OriginalLength* 字段。

定义了以下特殊值:

MQOL_UNDEFINED

未定义消息的原始长度。

OriginalLength 是 MQPUT 和 MQPUT1 调用上的输入字段, 但仅在特定情况下接受应用程序提供的值:

- 如果要放入的消息是段并且也是报告消息, 那么队列管理器将接受指定的值。该值必须为:
 - 如果分段不是最后一个分段, 那么大于零
 - 如果分段是最后一个分段, 那么不小于 0
 - 不小于消息中存在的长度

如果不满足这些条件, 那么调用将失败, 原因码为 MQRC_ORIGINAL_LENGTH_ERROR。

- 如果要放入的消息是段而不是报告消息, 那么队列管理器将忽略该字段并改为使用应用程序消息数据的长度。
- 在所有其他情况下, 队列管理器将忽略该字段并改为使用值 MQOL_UNDEFINED。

这是 MQGET 调用上的输出字段。

此字段的初始值为 MQOL_UNDEFINED。如果 *Version* 小于 MQMD_VERSION_2, 那么将忽略此字段。

Persistence (MQLONG)

这指示消息是否在队列管理器的系统故障和重新启动后仍然存在。对于 MQPUT 和 MQPUT1 调用, 值必须是下列其中一项:

MQPER_PERSISTENT

消息在队列管理器的系统故障和重新启动后仍然存在。一旦放入消息, 并且已落实放入该消息的工作单元 (如果将该消息作为工作单元的一部分放入), 那么将在辅助存储器上保留该消息。它将保留在该位置, 直到从队列中除去该消息, 并且已落实该消息所在的工作单元 (如果将该消息作为工作单元的一部分进行检索)。

将持久消息发送至远程队列时, 存储转发机制会将消息保存在路由至目标的每个队列管理器上, 直到已知消息已到达下一个队列管理器为止。

无法将持久消息放在以下位置:

- 临时动态队列数
- 映射到 CFLEVEL (\$TAG1) 或更低级别的 CFSTRUCT 对象的共享队列, 或者 CFSTRUCT 对象定义为 RECOVER (NO) 的共享队列。

持久消息可以放置在永久动态队列和预定义队列上。

MQPER_NOT_PERSISTENT

此消息通常不会在系统故障或队列管理器重新启动后继续存在。即使在队列管理器重新启动时在辅助存储器上找到消息的完整副本, 这也适用。

对于 NPMCLASS (HIGH) 队列, 非持久消息在正常队列管理器关闭并重新启动后仍然存在。

对于共享队列, 非持久消息会在队列共享组中的队列管理器重新启动后继续存在, 但不会在用于在共享队列上存储消息的耦合设施发生故障后继续存在。

MQPER_PERSISTENCE_AS_Q_DEF

- 如果队列是集群队列, 那么将从 *destination* 队列管理器中定义的 *DefPersistence* 属性获取消息的持久性, 该队列管理器拥有放置消息的队列的特定实例。通常, 集群队列的所有实例都具有相同的 *DefPersistence* 属性值, 尽管这不是强制的。

将消息放入目标队列时，会将 *DefPersistence* 的值复制到 *Persistence* 字段中。如果随后更改了 *DefPersistence*，那么已放置在队列上的消息不受影响。

- 如果队列不是集群队列，那么将从本地队列管理器上定义的 *DefPersistence* 属性获取消息的持久性，即使目标队列管理器是远程队列管理器也是如此。

如果队列名称解析路径中有多个定义，那么缺省持久性取自路径中第一个定义中此属性的值。这可以是：

- 别名队列
- 本地队列
- 远程队列的本地定义
- 队列管理器别名
- 传输队列 (例如，*DefXmitQName* 队列)

放入消息时，会将 *DefPersistence* 的值复制到 *Persistence* 字段中。如果随后更改了 *DefPersistence*，那么已放入的消息不受影响。

持久消息和非持久消息都可以存在于同一队列中。

在应答消息时，应用程序必须将请求消息的持久性用于应答消息。

对于 MQGET 调用，返回的值为 MQPER_PERSISTENT 或 MQPER_NOT_PERSISTENT。

这是 MQGET 调用的输出字段，也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的初始值为 MQPER_PERSISTENCE_AS_Q_DEF。

Priority (MQLONG)

对于 MQPUT 和 MQPUT1 调用，值必须大于或等于零；零是最低优先级。还可以使用以下特殊值：

MQPRI_PRIORITY_AS_Q_DEF

- 如果队列是集群队列，那么将从 *destination* 队列管理器中定义的 *DefPriority* 属性获取消息优先级，该队列管理器拥有放置消息的队列的特定实例。通常，集群队列的所有实例都具有相同的 *DefPriority* 属性值，尽管这不是强制的。

将消息放入目标队列时，会将 *DefPriority* 的值复制到 *Priority* 字段中。如果随后更改了 *DefPriority*，那么已放置在队列上的消息不受影响。

- 如果该队列不是集群队列，那么将从本地队列管理器中定义的 *DefPriority* 属性获取消息的优先级，即使目标队列管理器是远程队列管理器也是如此。

如果在队列名称解析路径中有多个定义，那么将从路径中的 *first* 定义中的此属性值获取缺省优先级。这可以是：

- 别名队列
- 本地队列
- 远程队列的本地定义
- 队列管理器别名
- 传输队列 (例如，*DefXmitQName* 队列)

放入消息时，会将 *DefPriority* 的值复制到 *Priority* 字段中。如果随后更改了 *DefPriority*，那么已放入的消息不受影响。

MQGET 调用返回的值始终大于或等于零；从不返回值 MQPRI_PRIORITY_AS_Q_DEF。

如果消息的优先级高于本地队列管理器支持的最大值 (此最大值由 *MaxPriority* queue-manager 属性提供)，那么队列管理器将接受该消息，但该消息将以队列管理器的最大优先级放在队列上；MQPUT 或 MQPUT1 调用将完成，并带有 MQCC_WARNING 和原因码 MQRC_PRIORITY_EXCEEDS_MAXIMUM。但是，*Priority* 字段保留由放置消息的应用程序指定的值。

在 z/OS 上，如果将带有 *MsgSeqNumber* of 1 的消息放入消息传递序列为 MQMDS_PRIORITY 且索引类型为 MQIT_GROUP_ID 的队列中，那么该队列可能会处理具有不同优先级的消息。如果将消息放在优先级为 0 或 1 的队列上，那么将处理该消息，就好像它具有优先级 2 一样。这是因为对此类型队列上的消息顺序进行了

优化，以启用高效的组完整性测试。有关消息传递序列 MQMDS_PRIORITY 和索引类型 MQIT_GROUP_ID 的更多信息，请参阅 `MsgDeliverySequence` 属性。

在回复消息时，应用程序必须将请求消息的优先级用于回复消息。在其他情况下，指定 MQPRI_PRIORITY_AS_Q_DEF 允许在不更改应用程序的情况下执行优先级调整。

这是 MQGET 调用的输出字段，也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的初始值为 MQPRI_PRIORITY_AS_Q_DEF。

PutAppl 名称 (MQCHAR28)

这是放置消息的应用程序的名称，并且是消息的源上下文的一部分。平台之间的内容不同，发行版之间的内容也可能不同。

有关消息上下文的更多信息，请参阅第 365 页的『MQMD 概述』和 [消息上下文](#)。

PutApplName 的格式取决于 *PutApplType* 的值，并且可以从一个发行版更改为另一个发行版。更改很少，但如果环境发生更改，那么会发生更改。

当队列管理器设置此字段 (即，对于除 MQPMO_SET_ALL_CONTEXT 以外的所有选项) 时，它会将此字段设置为由环境确定的值：

- 在 z/OS 上，队列管理器使用：
 - 对于 z/OS 批处理，这是 JES JOB 卡中的 8 字符作业名
 - 对于 TSO，7 字符的 TSO 用户标识
 - 对于 CICS，这是 8 字符的 applid，后跟 4 字符的 tranid
 - 对于 IMS，这是 8 字符 IMS 系统标识，后跟 8 字符的 PSB 名称
 - 对于 XCF，这是 8 字符的 XCF 组名，后跟 16 字符的 XCF 成员名
 - 对于队列管理器生成的消息，队列管理器名称的前 28 个字符
 - 对于没有 CICS 的分布式排队，通道启动程序的 8 字符作业名后跟放入死信队列的模块的 8 字符名称后跟 8 字符任务标识。

名称和字段的其余部分中的任何空格一样，每个名称都用空格填充在右边。如果有多个名称，那么它们之间没有分隔符。

- 在 Windows 系统上，队列管理器使用：
 - 对于 CICS 应用程序，这是 CICS 事务名称
 - 对于非 CICS 应用程序，可执行文件的标准名称的最右边 28 个字符
- 在 IBM i 上，队列管理器使用标准作业名。
- 在 UNIX 系统上，队列管理器使用：
 - 对于 CICS 应用程序，这是 CICS 事务名称
 - 对于非 CICS 应用程序，MQ 要求操作系统提供进程的名称。这是作为程序文件名返回的，没有完整路径。然后，MQ 将此进程名称放在 MQMD.PutApplName 字段如下所示：

AIX

如果名称小于或等于 28 个字节，那么将插入名称，并在右边填充空格。

如果名称大于 28 个字节，那么将插入名称的最左边的 28 个字节。

Linux 和 Solaris

如果名称小于或等于 15 个字节，那么将插入名称，并在右边填充空格。

如果名称大于 15 个字节，那么将插入名称的最左边的 15 个字节，并在右边填充空格。

HP-UX

如果名称小于或等于 14 个字节，那么将插入名称，并使用空格填充到右侧。

如果名称大于 14 个字节，那么将插入名称的最左边的 14 个字节，并在右边填充空格。

例如，如果运行 `/opt/mqm/samp/bin/amqsput QNAME QMNAME`，那么 *PutAppl* 名称为 'amqsput'。此 MQCHAR28 字段中有 21 个空格字符的填充。请注意，*PutAppl* 名称中不包含包含 `/opt/mqm/samp/bin` 的完整路径。

对于 MQPUT 和 MQPUT1 调用, 如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_ALL_CONTEXT, 那么这是输入/输出字段。将废弃字段中空字符后面的任何信息。队列管理器将空字符和任何后续字符转换为空白。如果未指定 MQPMO_SET_ALL_CONTEXT, 那么此字段将在输入时被忽略, 并且是仅输出字段。

PutApplType (MQLONG)

这是放置消息的应用程序类型, 并且是消息的 **源上下文** 的一部分。有关消息上下文的更多信息, 请参阅 [第 365 页的『MQMD 概述』](#) 和 [消息上下文](#)。

PutApplType 可以具有下列其中一种标准类型。您还可以定义自己的类型, 但只能使用 MQAT_USER_FIRST 到 MQAT_USER_LAST 范围内的值。

MQAT_AIX

AIX 应用程序 (与 MQAT_UNIX 的值相同)。

MQAT_BROKER

代理。

MQAT_CICS

CICS 事务。

MQAT_CICS_BRIDGE

CICS 网桥。

MQAT_CICS_VSE

CICS/VSE 事务。

MQAT_DOS

PC DOS 上的 WebSphere MQ MQI 客户机应用程序。

MQAT_DQM

分布式队列管理器代理程序。

MQAT_GUARDIAN

"同步守护程序" 应用程序 (与 MQAT_NSK 的值相同)。

MQAT_IMS

IMS 应用程序。

MQAT_IMS_BRIDGE

IMS 网桥。

MQAT_JAVA

Java。

MQAT_MVS

MVS 或 TSO 应用程序 (与 MQAT_ZOS 值相同)。

MQAT_NOTES_AGENT

Lotus Notes 代理程序应用程序。

MQAT_NSK

HP Integrity NonStop Server 应用程序。

MQAT_OS390

OS/390 应用程序 (与 MQAT_ZOS 的值相同)。

MQAT_OS400

IBM i 应用程序。

MQAT_QMGR

队列管理器。

MQAT_UNIX

UNIX 应用程序。

MQAT_VOS

Stratus VOS 应用程序。

MQAT_WINDOWS

16 位 Windows 应用程序。

MQAT_WINDOWS_NT

32 位 Windows 应用程序。

MQAT_WLM

z/OS 工作负载管理器应用程序。

MQAT_XCF

XCF。

MQAT_ZOS

z/OS 应用程序。

MQAT_DEFAULT

缺省应用程序类型。

这是运行应用程序的平台的缺省应用程序类型。

注: 此常量的值特定于环境。因此, 请始终使用适合于应用程序将在其上运行的平台的头, 包含或 COPY 文件来编译应用程序。

MQAT_UNKNOWN

使用此值来指示应用程序类型未知, 即使存在其他上下文信息也是如此。

MQAT_USER_FIRST

用户定义的应用程序类型的最小值。

MQAT_USER_LAST

用户定义的应用程序类型的最大值。

还可能出现以下特殊值:

MQAT_NO_CONTEXT

当放入没有上下文的消息时(即, 指定了 MQPMO_NO_CONTEXT 上下文选项), 队列管理器会设置此值。

检索消息时, 可以针对此值测试 *PutApplType* 以确定消息是否具有上下文(如果任何其他上下文字段非空白, 那么建议应用程序使用 MQPMO_SET_ALL_CONTEXT 将 *PutApplType* 从不设置为 MQAT_NO_CONTEXT)。

当队列管理器作为应用程序 put 的结果生成此信息时, 该字段将设置为由环境确定的值。在 IBM i 上, 它设置为 MQAT_OS400; 队列管理器从不使用 IBM i 上的 MQAT_CICS。

对于 MQPUT 和 MQPUT1 调用, 如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_ALL_CONTEXT, 那么这是输入/输出字段。如果未指定 MQPMO_SET_ALL_CONTEXT, 那么此字段将在输入时被忽略, 并且是仅输出字段。

这是 MQGET 调用的输出字段。此字段的初始值为 MQAT_NO_CONTEXT。

PutDate (MQCHAR8)

这是放入消息的日期, 并且是消息的 **源上下文** 的一部分。有关消息上下文的更多信息, 请参阅 [第 365 页的『MQMD 概述』](#) 和 [消息上下文](#)。

队列管理器生成此字段的日期所使用的格式为:

- YYYYMMDD

其中字符表示:

YYYY

年(四位数字)

MM

年月(01 至 12)

DD

月日(01 到 31)

格林威治标准时间(GMT)用于 *PutDate* 和 *PutTime* 字段, 前提是系统时钟精确设置为 GMT。

如果将消息作为工作单元的一部分放入, 那么日期是放入消息的时间, 而不是落实工作单元的日期。

对于 MQPUT 和 MQPUT1 调用，如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_ALL_CONTEXT，那么这是输入/输出字段。队列管理器不会检查该字段的内容，只是会废弃该字段中空字符之后的任何信息。队列管理器将空字符和任何后续字符转换为空白。如果未指定 MQPMO_SET_ALL_CONTEXT，那么此字段将在输入时被忽略，并且是仅输出字段。

这是 MQGET 调用的输出字段。此字段的长度由 MQ_PUT_DATE_LENGTH 给出。此字段的初始值是 C 中的空字符串，以及其他编程语言中的 8 空白字符。

PutTime (MQCHAR8)

这是放入消息的时间，并且是消息的源上下文的一部分。有关消息上下文的更多信息，请参阅第 365 页的『MQMD 概述』和消息上下文。

队列管理器生成此字段时使用的格式为：

- HHMMSSSTH

其中字符表示 (按顺序)：

HH

小时 (00 到 23)

MM

分钟 (00 到 59)

SS

秒 (00 到 59; 请参阅注释)

T

十分之一秒 (0 到 9)

H

百分之一秒 (0 到 9)

注：如果系统时钟同步到非常准确的时间标准，那么在极少数情况下，可能会在 *PutTime* 中返回 60 或 61 的秒数。当将闰秒插入全局时间标准时，会发生此情况。

格林威治标准时间 (GMT) 用于 *PutDate* 和 *PutTime* 字段，前提是系统时钟精确设置为 GMT。

如果将消息作为工作单元的一部分放置，那么时间是放置消息的时间，而不是落实工作单元的时间。

对于 MQPUT 和 MQPUT1 调用，如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_ALL_CONTEXT，那么这是输入/输出字段。队列管理器不会检查该字段的内容，只是会废弃该字段中空字符之后的任何信息。队列管理器将空字符和任何后续字符转换为空白。如果未指定 MQPMO_SET_ALL_CONTEXT，那么此字段将在输入时被忽略，并且是仅输出字段。

这是 MQGET 调用的输出字段。此字段的长度由 MQ_PUT_TIME_LENGTH 指定。此字段的初始值是 C 中的空字符串，以及其他编程语言中的 8 空白字符。

ReplyToQ (MQCHAR48)

这是发出消息获取请求的应用程序向其发送 MQMT_REPLY 和 MQMT_REPORT 消息的消息队列的名称。该名称是在由 *ReplyToQMgr* 标识的队列管理器上定义的队列的本地名称。此队列不得是模型队列，尽管发送队列管理器在放入消息时不会验证此队列。

对于 MQPUT 和 MQPUT1 调用，如果 *MsgType* 字段的值为 MQMT_REQUEST，或者如果 *Report* 字段请求任何报告消息，那么此字段不得为空。但是，指定 (或替换) 的值将传递到发出消息获取请求的应用程序，无论消息类型如何。

如果 *ReplyToQMgr* 字段为空，那么本地队列管理器将在其自己的队列定义中查找 *ReplyToQ* 名称。如果存在具有此名称的远程队列的本地定义，那么传输的消息中的 *ReplyToQ* 值将替换为远程队列定义中 *RemoteQName* 属性的值，当接收应用程序针对消息发出 MQGET 调用时，将在消息描述符中返回此值。如果远程队列的本地定义不存在，那么 *ReplyToQ* 保持不变。

如果指定了名称，那么它可以包含尾部空格；第一个空字符和后面的字符将被视为空格。否则，不会检查名称是否满足队列的命名规则；如果在传输的消息中替换了 *ReplyToQ*，那么对于传输的名称也是如此。所做的唯一检查是，如果情况需要，那么已指定名称。

如果不需要应答队列，请将 *ReplyToQ* 字段设置为空白，或者（在 C 编程语言中）设置为空字符串，或者设置为后跟空字符的一个或多个空白；不要使该字段未初始化。

对于 MQGET 调用，队列管理器始终将使用空白填充的名称返回到字段的长度。

如果需要报告消息的消息无法传递，并且报告消息也无法传递到指定的队列，那么原始消息和报告消息都将转至死信（未传递消息）队列（请参阅第 701 页的『队列管理器的属性』中描述的 *DeadLetterQName* 属性）。

这是 MQGET 调用的输出字段，也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的长度由 *MQ_Q_NAME_LENGTH* 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ReplyTo 队列管理器 (MQCHAR48)

这是要向其发送应答消息或报告消息的队列管理器的名称。*ReplyToQ* 是在此队列管理器上定义的队列的本地名称。

如果 *ReplyToQMgr* 字段为空，那么本地队列管理器将在其队列定义中查找 *ReplyToQ* 名称。如果存在具有此名称的远程队列的本地定义，那么传输的消息中的 *ReplyToQMgr* 值将替换为远程队列定义中 *RemoteQMgrName* 属性的值，当接收应用程序针对消息发出 MQGET 调用时，将在消息描述符中返回此值。如果远程队列的本地定义不存在，那么随消息一起传输的 *ReplyToQMgr* 是本地队列管理器的名称。

如果指定了名称，那么它可以包含尾部空格；第一个空字符和后面的字符将被视为空格。否则，不会检查名称是否满足队列管理器的命名规则，或者此名称对于发送队列管理器是否已知；如果在传输的消息中替换了 *ReplyToQMgr*，那么对于传输的名称也是如此。

如果不需要应答队列，请将 *ReplyToQMgr* 字段设置为空白，或者（在 C 编程语言中）设置为空字符串，或者设置为后跟空字符的一个或多个空白；不要使该字段未初始化。

对于 MQGET 调用，队列管理器始终将使用空白填充的名称返回到字段的长度。

这是 MQGET 调用的输出字段，也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的长度由 *MQ_Q_MGR_NAME_LENGTH* 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

Report (MQLONG)

报告消息是关于另一条消息的消息，用于通知应用程序与原始消息相关的预期事件或意外事件。*Report* 字段使发送原始消息的应用程序能够指定需要哪些报告消息，是否要将应用程序消息数据包括在这些消息中，以及（对于报告和应答）如何设置报告或应答消息中的消息和相关标识。可以请求以下类型的任何或全部（或无）报告消息：

- 异常
- 到期
- 到达时确认 (COA)
- 交付时确认 (COD)
- 肯定操作通知 (PAN)
- 否定操作通知 (NAN)

如果需要多种类型的报告消息，或者需要其他报告选项，那么值可以是：

- 一起添加（请勿多次添加同一常量），或者
- 通过按位“或”运算组合在一起（如果编程语言支持位运算）。

接收报告消息的应用程序可以通过检查 MQMD 中的 *Feedback* 字段来确定生成报告的原因；请参阅 *Feedback* 字段以获取更多详细信息。

将消息放入主题时使用报告选项可能会导致生成零条，一条或多条报告消息并将其发送到应用程序。这是因为发布消息可能发送到零个，一个或多个预订应用程序。

异常选项：指定列出的其中一个选项以请求异常报告消息。

MQRO_EXCEPTION

当消息发送到另一个队列管理器并且无法将消息传递到指定的目标队列时，消息通道代理将生成此类型的报告。例如，目标队列或中间传输队列可能已满，或者消息可能对于队列过大。

异常报告消息的生成取决于原始消息的持久性，以及原始消息通过的消息通道（正常或快速）的速度：

- 对于所有持久消息以及通过正常消息通道传输的非持久消息，如果发送应用程序针对错误情况指定的操作可以成功完成，那么将仅生成异常报告。发送应用程序可以指定下列其中一项操作，以在出现错误情况时控制原始消息的处置：

- MQRO_DEAD_LETTER_Q (将原始消息放在死信队列上)。
- MQRO_DISCARD_MSG (这将废弃原始消息)。

如果无法成功完成发送应用程序指定的操作，那么会将原始消息保留在传输队列上，并且不会生成异常报告消息。

- 对于通过快速消息通道传输的非持久消息，将从传输队列中除去原始消息，并且即使无法成功完成针对错误情况的指定操作，也会生成异常报告。例如，如果指定了 MQRO_DEAD_LETTER_Q，但无法将原始消息放在死信队列上，因为该队列已满，将生成异常报告消息并废弃原始消息。

有关正常和快速消息通道的更多信息，请参阅 [非持久消息速度 \(NPMSPEED\)](#)。

如果放置原始消息的应用程序可以通过 MQPUT 或 MQPUT1 调用返回的原因码同步通知问题，那么不会生成异常报告。

应用程序还可以发送异常报告，以指示无法处理消息（例如，由于是借记交易会超出其贷记限额）。

报告消息不包含来自原始消息的消息数据。

请勿指定 MQRO_EXCEPTION，MQRO_EXCEPTION_WITH_DATA 和 MQRO_EXCEPTION_WITH_FULL_DATA 中的多个。

MQRO_EXCEPTION_WITH_DATA

这与 MQRO_EXCEPTION 相同，只是原始消息中的应用程序消息数据的前 100 个字节包含在报告消息中。如果原始消息包含一个或多个 MQ 头结构，那么除了 100 字节的应用程序数据外，还会将它们包括在报告消息中。

请勿指定 MQRO_EXCEPTION，MQRO_EXCEPTION_WITH_DATA 和 MQRO_EXCEPTION_WITH_FULL_DATA 中的多个。

MQRO_EXCEPTION_WITH_FULL_DATA

需要完整数据的异常报告。

这与 MQRO_EXCEPTION 相同，只是原始消息中的所有应用程序消息数据都包含在报告消息中。

请勿指定 MQRO_EXCEPTION，MQRO_EXCEPTION_WITH_DATA 和 MQRO_EXCEPTION_WITH_FULL_DATA 中的多个。

到期选项: 指定列出的其中一个选项以请求到期报告消息。

MQRO_EXPIRATION

此类型的报告由队列管理器生成，前提是消息在传递到应用程序之前被废弃，因为其到期时间已过（请参阅 *Expiry* 字段）。如果未设置此选项，那么如果由于此原因而废弃消息（即使指定了其中一个 MQRO_EXCEPTION_* 选项），也不会生成报告消息。

报告消息不包含来自原始消息的消息数据。

请勿指定 MQRO_EXPIRATION，MQRO_EXPIRATION_WITH_DATA 和 MQRO_EXPIRATION_WITH_FULL_DATA 中的多个。

MQRO_EXPIRATION_WITH_DATA

这与 MQRO_EXPIRATION 相同，只是原始消息中的应用程序消息数据的前 100 个字节包含在报告消息中。如果原始消息包含一个或多个 MQ 头结构，那么除了 100 字节的应用程序数据外，还会将它们包括在报告消息中。

请勿指定 MQRO_EXPIRATION，MQRO_EXPIRATION_WITH_DATA 和 MQRO_EXPIRATION_WITH_FULL_DATA 中的多个。

MQRO_EXPIRATION_WITH_FULL_DATA

这与 MQRO_EXPIRATION 相同，只是原始消息中的所有应用程序消息数据都包含在报告消息中。

请勿指定 MQRO_EXPIRATION，MQRO_EXPIRATION_WITH_DATA 和 MQRO_EXPIRATION_WITH_FULL_DATA 中的多个。

到达时确认选项: 指定列出的其中一个选项以请求到达时确认报告消息。

MQRO_COA

此类型的报告由拥有目标队列的队列管理器在将消息放入目标队列时生成。报告消息不包含来自原始消息的消息数据。

如果将消息作为工作单元的一部分放入，并且目标队列是本地队列，那么仅当落实工作单元时，才能检索队列管理器生成的 COA 报告消息。

如果消息描述符中的 *Format* 字段为 MQFMT_XMIT_Q_HEADER 或 MQFMT_DEAD_LETTER_HEADER，那么不会生成 COA 报告。如果将消息放在传输队列上，或者无法传递消息并将其放在死信队列上，那么这将阻止生成 COA 报告。

对于 IMS 网桥队列，当消息到达 IMS 队列 (从 IMS 接收的应答) 时，而不是将消息放入 MQ 网桥队列时，将生成 COA 报告。这意味着如果 IMS 未处于活动状态，那么在启动 IMS 并在 IMS 队列上排队消息之前不会生成 COA 报告。

运行使用 MQMD.Report= MQRO_COA 必须对应答队列具有 + 钝化权限。如果用户没有 + 钝化权限，那么 COA 报告消息无法到达应答队列。尝试将报告消息放在死信队列上。

请勿指定多个 MQRO_COA，MQRO_COA_WITH_DATA 和 MQRO_COA_WITH_FULL_DATA。

MQRO_COA_WITH_DATA

这与 MQRO_COA 相同，只是原始消息中的应用程序消息数据的前 100 个字节包含在报告消息中。如果原始消息包含一个或多个 MQ 头结构，那么除了 100 字节的应用程序数据外，还会将它们包括在报告消息中。

请勿指定多个 MQRO_COA，MQRO_COA_WITH_DATA 和 MQRO_COA_WITH_FULL_DATA。

MQRO_COA_WITH_FULL_DATA

这与 MQRO_COA 相同，只是原始消息中的所有应用程序消息数据都包含在报告消息中。

请勿指定多个 MQRO_COA，MQRO_COA_WITH_DATA 和 MQRO_COA_WITH_FULL_DATA。

传递时确认选项: 指定列出的其中一个选项以请求传递时确认报告消息。

MQRO_COD

当应用程序以从队列中删除消息的方式从目标队列中检索消息时，队列管理器将生成此类型的报告。报告消息不包含来自原始消息的消息数据。

如果将消息作为工作单元的一部分进行检索，那么将在同一工作单元中生成报告消息，以便在落实工作单元之前报告不可用。如果工作单元回退，那么不会发送报告。

如果使用 MQGMO_MARK_SKIP_BACKOUT 选项检索消息，那么不会始终生成 COD 报告。如果回退了主工作单元，但落实了辅助工作单元，那么将从队列中除去消息，但不会生成 COD 报告。

如果消息描述符中的 *Format* 字段为 MQFMT_DEAD_LETTER_HEADER，那么不会生成 COD 报告。如果无法传递消息并将其放入死信队列中，那么这将阻止生成 COD 报告。

如果目标队列是 XCF 队列，那么 MQRO_COD 无效。

请勿指定多个 MQRO_COD，MQRO_COD_WITH_DATA 和 MQRO_COD_WITH_FULL_DATA。

MQRO_COD_WITH_DATA

这与 MQRO_COD 相同，只是原始消息中的应用程序消息数据的前 100 个字节包含在报告消息中。如果原始消息包含一个或多个 MQ 头结构，那么除了 100 字节的应用程序数据外，还会将它们包括在报告消息中。

如果在原始消息的 MQGET 调用上指定了 MQGMO_ACCEPT_TRUNCATED_MSG，并且检索到的消息被截断，那么放入报告消息中的应用程序消息数据量取决于环境：

- 在 z/OS 上，最小值为：

- 原始消息的长度
- 用于检索消息的缓冲区的长度
- 100 字节。
- 在其他环境中，它是以下值的最小值：
 - 原始消息的长度
 - 100 字节。

如果目标队列是 XCF 队列，那么 MQRO_COD_WITH_DATA 无效。

请勿指定多个 MQRO_COD，MQRO_COD_WITH_DATA 和 MQRO_COD_WITH_FULL_DATA。

MQRO_COD_WITH_FULL_DATA

这与 MQRO_COD 相同，只是原始消息中的所有应用程序消息数据都包含在报告消息中。

如果目标队列是 XCF 队列，那么 MQRO_COD_WITH_FULL_DATA 无效。

请勿指定多个 MQRO_COD，MQRO_COD_WITH_DATA 和 MQRO_COD_WITH_FULL_DATA。

操作-通知选项: 指定列出的一个或两个选项，以请求接收应用程序发送正面操作或负面操作报告消息。

MQRO_PAN

此类型的报告由检索消息并对其执行操作的应用程序生成。它指示已成功执行消息中请求的操作。生成报告的应用程序确定是否要将任何数据包含在报告中。

除了将此请求传输到检索消息的应用程序之外，队列管理器不会根据此选项执行任何操作。如果需要，检索应用程序必须生成报告。

MQRO_NAN

此类型的报告由检索消息并对其执行操作的应用程序生成。它指示未成功执行消息中请求的操作。生成报告的应用程序确定是否要将任何数据包含在报告中。例如，您可能希望包含一些数据，以指示无法执行请求的原因。

除了将此请求传输到检索消息的应用程序之外，队列管理器不会根据此选项执行任何操作。如果需要，检索应用程序必须生成报告。

应用程序必须确定哪些条件对应于肯定操作，哪些条件对应于否定操作。但是，如果仅部分执行了请求，请生成 NAN 报告，而不是 PAN 报告 (如果请求)。每个可能的条件都必须对应于肯定操作或否定操作，但不能同时对应于两者。

消息标识选项: 指定列出的其中一个选项，以控制如何设置报告消息 (或应答消息) 的 *MsgId*。

MQRO_NEW_MSG_ID

这是缺省操作，指示如果由于此消息而生成报告或应答，那么将为报告或应答消息生成新的 *MsgId*。

MQRO_PASS_MSG_ID

如果由于此消息而生成报告或应答，那么会将此消息的 *MsgId* 复制到报告或应答消息的 *MsgId*。

对于接收发布副本的每个订户，发布消息的 *MsgId* 将不同，因此复制到报告或回复消息中的 *MsgId* 将不同。

如果未指定此选项，那么将采用 MQRO_NEW_MSG_ID。

相关标识选项: 指定列出的其中一个选项，以控制如何设置报告消息 (或应答消息) 的 *CorrelId*。

MQRO_COPY_MSG_ID_TO_CORREL_ID

这是缺省操作，指示如果由于此消息而生成报告或应答，那么会将此消息的 *MsgId* 复制到报告或应答消息的 *CorrelId*。

对于接收发布副本的每个订户，发布消息的 *MsgId* 将不同，因此复制到报告或回复消息的 *CorrelId* 中的 *MsgId* 将不同。

MQRO_PASS_CORREL_ID

如果由于此消息而生成报告或应答，那么会将此消息的 *CorrelId* 复制到报告或应答消息的 *CorrelId*。

发布消息的 *CorrelId* 将特定于订户，除非它使用 `MQSO_SET_CORREL_ID` 选项并将 `MQSD` 中的 `SubCorrelId` 字段设置为 `MQCI_NONE`。因此，复制到报告或应答消息的 *CorrelId* 中的 *CorrelId* 可能对每个报告或应答消息都不同。

如果未指定此选项，那么将采用 `MQRO_COPY_MSG_ID_TO_CORREL_ID`。

回复请求或生成报告消息的服务器必须检查是否在原始消息中设置了 `MQRO_PASS_MSG_ID` 或 `MQRO_PASS_CORREL_ID` 选项。如果是，那么服务器必须执行针对这些选项描述的操作。如果两者都未设置，那么服务器必须执行相应的缺省操作。

处置选项: 指定列出的其中一个选项，以在原始消息无法传递到目标队列时控制其处置。应用程序可以独立于请求异常报告来设置处置选项。

MQRO_DEAD_LETTER_Q

这是缺省操作，如果无法将消息传递到目标队列，那么将消息放在死信队列上。在以下情况下会发生此情况：

- 当放置原始消息的应用程序无法通过 `MQPUT` 或 `MQPUT1` 调用返回的原因码同步通知问题时。如果发送方请求了异常报告消息，那么将生成异常报告消息。
- 将原始消息放入主题的应用程序

MQRO_DISCARD_MSG

如果无法将消息传递到目标队列，那么将废弃该消息。在以下情况下会发生此情况：

- 当放置原始消息的应用程序无法通过 `MQPUT` 或 `MQPUT1` 调用返回的原因码同步通知问题时。如果发送方请求了异常报告消息，那么将生成异常报告消息。
- 将原始消息放入主题的应用程序

如果要将原始消息返回给发送方，而未将原始消息放在死信队列上，那么发送方必须使用 `MQRO_EXCEPTION_WITH_FULL_DATA` 指定 `MQRO_DISCARD_MSG`。

MQRO_PASS_DISCARD_AND_EXPIRY

如果对消息设置了此选项，并且由于此选项而生成了报告或应答，那么该报告的消息描述符将继承：

- `MQRO_DISCARD_MSG` (如果已设置)。
- 消息的剩余到期时间 (如果这不是到期报告)。如果这是到期报告，那么到期时间设置为 60 秒。

"活动" 选项

MQRO_ACTIVITY

使用此值允许在整个队列管理器网络中跟踪 **any** 消息的路由。可以在任何当前用户消息上指定报告选项，从而立即允许您开始计算消息通过网络的路由。

如果生成消息的应用程序无法打开活动报告，那么可以使用队列管理器管理员提供的 API 交叉出口来打开报告。

注:

1. 网络中能够生成活动报告的队列管理器越少，路由的详细程度越低。
2. 活动报告可能难以以正确的顺序确定所采用的路线。
3. 活动报告可能无法找到指向其请求的目标的路径。
4. 任何队列管理器都必须接受具有此报告选项集的消息，即使它们不了解该选项也是如此。这允许对任何用户消息设置报告选项，即使这些用户消息由非 V 6.0 或更高版本的队列管理器处理也是如此。
5. 如果某个进程 (队列管理器或用户进程) 使用此选项集对消息执行活动，那么可以选择生成并放置活动报告。

缺省选项: 如果不需要报告选项，请指定以下内容：

MQRO_NONE

使用此值可指示未指定任何其他选项。 `MQRO_NONE` 定义为帮助程序文档。不打算将此选项与任何其他选项一起使用，但由于其值为零，因此无法检测到此类使用。

常用信息:

1. 发送原始消息的应用程序必须特别请求所需的所有报告类型。例如，如果请求 COA 报告但未请求异常报告，那么在将消息放入目标队列时会生成 COA 报告，但如果消息到达目标队列时目标队列已满，那么不会生成异常报告。如果未设置 *Report* 选项，那么队列管理器或消息通道代理 (MCA) 不会生成任何报告消息。

即使本地队列管理器无法识别某些报告选项，也可以指定这些报告选项；当 *destination* 队列管理器要处理该选项时，这很有用。请参阅第 788 页的『报告选项和消息标志』以获取更多详细信息。

如果请求了报告消息，那么必须在 *ReplyToQ* 字段中指定要将报告发送到的队列的名称。接收到报告消息时，可以通过检查消息描述符中的 *Feedback* 字段来确定报告的性质。

2. 如果生成报告消息的队列管理器或 MCA 无法将报告消息放在应答队列上 (例如，因为应答队列或传输队列已满)，那么会将报告消息改为放在死信队列上。如果该也失败，或者没有死信队列，那么执行的操作取决于报告消息的类型：

- 如果报告消息是异常报告，那么生成异常报告的消息将保留在其传输队列上；这将确保消息不会丢失。
- 对于所有其他报告类型，将废弃报告消息并正常继续处理。执行此操作的原因是原始消息已安全传递 (对于 COA 或 COD 报告消息)，或者不再感兴趣 (对于到期报告消息)。

将报告消息成功放入队列 (目标队列或中间传输队列) 后，该消息将不再接受特殊处理；它将与其他消息一样受到处理。

3. 生成报告时，将打开 *ReplyToQ* 队列，并使用 *UserIdentifier* 的权限将报告消息放入导致报告的消息的 MQMD 中，以下情况除外：

- 接收 MCA 生成的异常报告将与 MCA 在尝试放置导致报告的消息时使用的任何权限放在一起。
- 将由队列管理器生成的 COA 报告放在生成报告的队列管理器上时，无论使用何种权限，都会导致该报告的消息放在该队列管理器上。例如，如果消息是由接收 MCA 使用 MCA 的用户标识放入的，那么队列管理器将使用 MCA 的用户标识放入 COA 报告。

生成报告的应用程序必须使用与生成应答相同的权限；这通常是原始消息中用户标识的权限。

如果报告必须传递到远程目标，那么发件人和接收方可以决定是否接受该报告，其方式与他们对其他消息的方式相同。

4. 如果请求包含数据的报告消息：

- 始终使用原始消息的发送方所请求的数据量来生成报告消息。如果报告消息对于应答队列太大，那么将进行上述处理；报告消息从不被截断以适合应答队列。
- 如果原始消息的 *Format* 是 MQFMT_XMIT_Q_HEADER，那么报告中包含的数据不包含 MQXQH。报告数据从原始消息中 MQXQH 之外的数据的第一个字节开始。无论队列是否为传输队列，都会发生此情况。

5. 如果在应答队列中接收到 COA，COD 或到期报告消息，那么将保证原始消息已到达，已传递或已到期 (视情况而定)。但是，如果请求了这些报告消息中的一条或多条，并且未接收到这些报告消息，那么无法假定发生反向操作，因为可能发生了以下情况之一：

- a. 报告消息已挂起，因为链接已关闭。
- b. 由于中间传输队列或应答队列中存在阻塞情况 (例如，队列已满或禁止放入)，因此报告消息被挂起。
- c. 报告消息位于死信队列上。
- d. 当队列管理器尝试生成报告消息时，它既不能将其放在相应的队列上，也不能放在死信队列上，因此无法生成报告消息。
- e. 在要报告的操作 (到达，传递或到期) 与生成相应的报告消息之间，队列管理器发生了故障。(如果应用程序在工作单元中检索原始消息，那么 COD 报告消息不会发生此情况，因为 COD 报告消息是在同一工作单元中生成的。)

由于以上原因 1，2 和 3，异常报告消息可以以相同方式挂起。但是，当 MCA 无法生成异常报告消息 (报告消息不能放在应答队列或死信队列上) 时，原始消息将保留在发送方的传输队列上，并且通道已关闭。无论报告消息是在通道的发送端还是接收端生成，都会发生此情况。

6. 如果临时阻塞原始消息 (导致生成异常报告消息并将原始消息放入死信队列)，但阻塞清除，然后应用程序从死信队列读取原始消息并将其再次放入其目标，那么可能会发生以下情况：

- 即使生成了异常报告消息，原始消息也最终成功到达其目标。

- 针对单个原始消息生成了多条异常报告消息，因为该原始消息稍后可能会迂到另一个阻塞。

放入主题时的报告消息:

1. 将消息放入主题时，可以生成报告。此消息将发送到主题的所有订户，这些订户可以是零，一个或多个。在选择使用报告选项时，应该考虑这一点，因为可能会生成大量报告消息。
2. 将消息放入主题时，可能有许多要为其提供消息副本的目标队列。如果其中一些目标队列存在问题 (例如队列已满)，那么 MQPUT 的成功完成取决于 NPMMSGDLV 或 PMSGDLV 的设置 (取决于消息的持久性)。如果设置为必须成功将消息传递到目标队列 (例如，它是持久订户的持久消息，并且 PMSGDLV 设置为 ALL 或 ALLDUR)，那么成功将定义为满足以下条件之一:
 - 成功放入订户队列
 - 如果订户队列无法获取消息，请使用 MQRO_DEAD_LETTER_Q 并成功将其放入死信队列
 - 如果订户队列无法获取消息，请使用 MQRO_DISCARD_MSG。

报告消息段的的消息:

1. 可以为允许分段的消息请求报告消息 (请参阅 MQMF_SEGMENTATION_ALLOWED 标志的描述)。如果队列管理器发现需要对消息进行分段，那么可以为随后迂到相关情况的每个分段生成报告消息。应用程序必须准备好接收针对所请求的每种类型的报告消息的多条报告消息。使用报告消息中的 *GroupId* 字段将多个报告与原始消息的组标识相关联，*Feedback* 字段标识每个报告消息的类型。
2. 如果 MQGMO_LOGICAL_ORDER 用于检索分段的报告消息，请注意后续 MQGET 调用可能会返回不同类型的报告。例如，如果针对队列管理器分段的消息同时请求 COA 和 COD 报告，那么针对报告消息的 MQGET 调用可能会以不可预测的方式返回 COA 和 COD 报告消息。通过使用 MQGMO_COMPLETE_MSG 选项 (可以选择使用 MQGMO_ACCEPT_TRUNCATED_MSG) 来避免此情况。MQGMO_COMPLETE_MSG 导致队列管理器重新组合具有相同报告类型的报告消息。例如，第一个 MQGET 调用可能会重新组合所有与原始消息相关的 COA 消息，而第二个 MQGET 调用可能会重新组合所有 COD 消息。首先重新组合的内容取决于首先在队列上出现的报告消息类型。
3. 自己放置分段的应用程序可以为每个分段指定不同的报告选项。但是，请注意以下几点:
 - 如果使用 MQGMO_COMPLETE_MSG 选项检索段，那么队列管理器仅采用第一个段中的报告选项。
 - 如果逐个检索段，并且其中大多数段都有一个 MQRO_COD_* 选项，但至少有一个段没有，那么不能使用 MQGMO_COMPLETE_MSG 选项通过单个 MQGET 调用来检索报告消息，或者使用 MQGMO_ALL_SEGMENTS_AVAILABLE 选项来检测所有报告消息何时到达。
4. 在 MQ 网络中，队列管理器可以具有不同的功能。如果段的报告消息是由不支持分段的队列管理器或 MCA 生成的，那么缺省情况下，队列管理器或 MCA 不会在报告消息中包含必需的段信息，这可能导致难以识别导致生成报告的原始消息。通过使用报告消息 (即，通过指定相应的 MQRO_*_WITH_DATA 或 MQRO_*_WITH_FULL_DATA 选项) 请求数据来避免此困难。但是，请注意，如果指定了 MQRO_*_WITH_DATA，那么如果报告消息由不支持分段的队列管理器或 MCA 生成，那么可能会将少于 个字节的程序消息数据返回到检索报告消息的应用程序。

报告消息的消息描述符的内容: 当队列管理器或消息通道代理 (MCA) 生成报告消息时，它会将消息描述符中的字段设置为以下值，然后以正常方式放置消息。

| MQMD 中的字段 | 使用的值 |
|-----------------------|---|
| <i>StrucId</i> | MQMD_STRUC_ID |
| <i>Version</i> | MQMD_VERSION_2 |
| <i>Report</i> | MQRO_NONE |
| <i>MsgType</i> | MQMT_REPORT |
| <i>Expiry</i> | MQEI_UNLIMITED |
| <i>Feedback</i> | 适合于报告的性质 (MQFB_COA, MQFB_COD, MQFB_EXPIRATION 或 MQRC_* 值) |
| <i>Encoding</i> | 已从原始消息描述符复制 |
| <i>CodedCharSetId</i> | 已从原始消息描述符复制 |

MQMD 中的字段

使用的值

| | |
|-------------------------|---|
| <i>Format</i> | 已从原始消息描述符复制 |
| <i>Priority</i> | 已从原始消息描述符复制 |
| <i>Persistence</i> | 已从原始消息描述符复制 |
| <i>MsgId</i> | 由原始消息描述符中的报告选项指定 |
| <i>CorrelId</i> | 由原始消息描述符中的报告选项指定 |
| <i>BackoutCount</i> | 0 |
| <i>ReplyToQ</i> | 空白 |
| <i>ReplyToQMGr</i> | 队列管理器的名称 |
| <i>UserIdentifier</i> | 由 MQPMO_PASS_IDENTITY_CONTEXT 选项设置 |
| <i>AccountingToken</i> | 由 MQPMO_PASS_IDENTITY_CONTEXT 选项设置 |
| <i>ApplIdentityData</i> | 由 MQPMO_PASS_IDENTITY_CONTEXT 选项设置 |
| <i>PutApplType</i> | MQAT_QMGR, 或者适用于消息通道代理程序 |
| <i>PutApplName</i> | 队列管理器名称或消息通道代理程序名称的前 28 个字节。对于由 IMS 网桥生成的报告消息, 此字段包含与消息相关的 IMS 系统的 XCF 组名和 XCF 成员名。 |
| <i>PutDate</i> | 发送报告消息的日期 |
| <i>PutTime</i> | 发送报告消息的时间 |
| <i>ApplOriginData</i> | 空白 |
| <i>GroupId</i> | 已从原始消息描述符复制 |
| <i>MsgSeqNumber</i> | 已从原始消息描述符复制 |
| <i>Offset</i> | 已从原始消息描述符复制 |
| <i>MsgFlags</i> | 已从原始消息描述符复制 |
| <i>OriginalLength</i> | 如果不是 MQOL_UNDEFINED, 那么从原始消息描述符复制, 并设置为原始消息数据的长度, 否则 |

建议生成报告的应用程序设置类似的值, 但以下值除外:

- *ReplyToQMGr* 字段可以设置为空白 (放入消息时, 队列管理器会将此字段更改为本地队列管理器的名称)。
- 使用将用于应答的选项 (通常为 MQPMO_PASS_IDENTITY_CONTEXT) 来设置上下文字段。

分析报告字段: *Report* 字段包含子字段; 因此, 需要检查请求特定报告的消息的发送方是否必须使用 [第 789 页的『分析报告字段』](#) 中描述的其中一种方法的应用程序。

这是 MQGET 调用的输出字段, 也是 MQPUT 和 MQPUT1 调用的输入字段。此字段的初始值为 MQRO_NONE。

StrucId (MQCHAR4)

这是结构标识, 必须是:

MQMD_STRUC_ID

消息描述符结构的标识。

对于 C 编程语言, 还定义了常量 MQMD_STRUC_ID_ARRAY; 此值与 MQMD_STRUC_ID 相同, 但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQMD_STRUC_ID。

UserIdentifier (MQCHAR12)

这是消息的 **身份上下文** 的一部分。有关消息上下文的更多信息，请参阅 [第 365 页的『MQMD 概述』](#) 和 [消息上下文](#)。

UserIdentifier 指定发起消息的应用程序的用户标识。队列管理器将此信息视为字符数据，但不定义其格式。

接收到消息后，请在后续 MQOPEN 或 MQPUT1 调用的 *ObjDesc* 参数的 *AlternateUserId* 字段中使用 *UserIdentifier*，以对 *UserIdentifier* 用户而不是执行打开操作的应用程序执行授权检查。

当队列管理器为 MQPUT 或 MQPUT1 调用生成此信息时：

- 在 z/OS 上，如果指定了 MQOO_ALTERNATE_USER_AUTHORITY 或 MQPMO_ALTERNATE_USER_AUTHORITY 选项，那么队列管理器将使用 MQOPEN 或 MQPUT1 调用的 *ObjDesc* 参数中的 *AlternateUserId*。如果未指定相关选项，那么队列管理器将使用从环境中确定的用户标识。
- 在其他环境中，队列管理器始终使用根据环境确定的用户标识。

从环境中确定用户标识时：

- 在 z/OS 上，队列管理器使用：
 - 对于 MVS (批处理)，这是 JES JOB 卡或启动式任务中的用户标识
 - 对于 TSO，在作业提交期间传播到作业的用户标识
 - 对于 CICS，这是与任务关联的用户标识
 - 对于 IMS，用户标识取决于应用程序类型：

- 针对：

- 非消息 BMP 区域
- 非消息 IFP 区域
- 未发出成功 GU 调用的消息 BMP 和消息 IFP 区域

队列管理器使用区域 JES JOB 卡中的用户标识或 TSO 用户标识。如果这些值为空白或空，那么它将使用程序规范块 (PSB) 的名称。

- 针对：

- 已发出成功 GU 调用的消息 BMP 和消息 IFP 区域
- MPP 区域

队列管理器使用下列其中一项：

- 与消息关联的注册用户标识
- 逻辑终端 (LTERM) 名称
- 区域 JES JOB 卡中的用户标识
- TSO 用户标识
- PSB 名称

- 在 IBM i 上，队列管理器使用与应用程序作业关联的用户概要文件的名称。
- 在 UNIX 系统上，队列管理器使用：
 - 应用程序的登录名
 - 进程的有效用户标识 (如果没有可用的登录)
 - 与事务关联的用户标识 (如果应用程序是 CICS 事务)
- 在 Windows 系统上，队列管理器使用已登录用户名的前 12 个字符。

此字段通常是队列管理器生成的输出字段，但对于 MQPUT 或 MQPUT1 调用，您可以使此字段成为输入/输出字段并指定 *UserIdentification* 字段，而不是让队列管理器生成此信息。如果不希望队列管理器为 MQPUT 或 MQPUT1 调用生成 *UserIdentifier* 字段，请在 *PutMsgOpts* 参数中指定

MQPMO_SET_IDENTITY_CONTEXT 或 MQPMO_SET_ALL_CONTEXT，并在 *UserIdentifier* 字段中指定用户标识。

对于 MQPUT 和 MQPUT1 调用，如果在 *PutMsgOpts* 参数中指定了 MQPMO_SET_IDENTITY_CONTEXT 或 MQPMO_SET_ALL_CONTEXT，那么这是输入/输出字段。将废弃字段中空字符后面的任何信息。队列管理器将空字符和任何后续字符转换为空白。如果未指定 MQPMO_SET_IDENTITY_CONTEXT 或 MQPMO_SET_ALL_CONTEXT，那么此字段在输入时将被忽略，并且是仅输出字段。

成功完成 MQPUT 或 MQPUT1 调用后，此字段包含随消息一起传输的 *UserIdentifier* (如果将其放入队列)。如果保留消息，那么这将是 *UserIdentifier* 的值 (请参阅 MQPMO_RETAIN 的描述以获取有关保留发布的更多详细信息)，但在将消息作为发布内容发送给订户时，不会将此值用作 *UserIdentifier*，因为它们提供的值将覆盖发送给订户的所有发布内容中的 *UserIdentifier*。如果消息没有上下文，那么该字段完全为空白。

这是 MQGET 调用的输出字段。此字段的长度由 MQ_USER_ID_LENGTH 给出。此字段的初始值是 C 中的空字符串，以及其他编程语言中的 12 个空白字符。

Version (MQLONG)

这是结构版本号，必须是下列其中一项：

MQMD_VERSION_1

Version-1 消息描述符结构。

此版本在所有环境中都受支持。

MQMD_VERSION_2

Version-2 消息描述符结构。

此版本在所有 WebSphere MQ V6.0 和更高版本环境以及连接到这些系统的 WebSphere MQ MQI 客户机中受支持。

注：使用 version-2 MQMD 时，队列管理器将对应用程序消息数据开头可能存在的任何 MQ 头结构执行其他检查；有关更多详细信息，请参阅 MQPUT 调用的使用说明。

仅在结构的最新版本中存在的字段在字段描述中标识为此类字段。以下常量指定当前版本的版本号：

MQMD_CURRENT_VERSION

消息描述符结构的当前版本。

这始终是一个输入字段。此字段的初始值为 MQMD_VERSION_1。

MQMD 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|-----------------------|----------------------------|-------|
| <i>StrucId</i> | MQMD_STRUC_ID | 'MD' |
| <i>Version</i> | MQMD_VERSION_1 | 1 |
| <i>Report</i> | MQRO_NONE | 0 |
| <i>MsgType</i> | MQMT_DATAGRAM | 8 |
| <i>Expiry</i> | MQEI_UNLIMITED | -1 |
| <i>Feedback</i> | MQFB_NONE | 0 |
| <i>Encoding</i> | MQENC_NATIVE | 取决于环境 |
| <i>CodedCharSetId</i> | MQCCSI_Q_MGR | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Priority</i> | MQPRI_PRIORITY_AS_Q_DEF | -1 |
| <i>Persistence</i> | MQPER_PERSISTENCE_AS_Q_DEF | 2 |

表 515: MQMD 的 MQMD 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|-------------------------|-----------------|---------|
| <i>MsgId</i> | MQMI_NONE | Null |
| <i>CorrelId</i> | MQCI_NONE | Null |
| <i>BackoutCount</i> | None | 0 |
| <i>ReplyToQ</i> | None | 空字符串或空白 |
| <i>ReplyToQMgr</i> | None | 空字符串或空白 |
| <i>UserIdentifier</i> | None | 空字符串或空白 |
| <i>AccountingToken</i> | MQACT_NONE | Null |
| <i>ApplIdentityData</i> | None | 空字符串或空白 |
| <i>PutApplType</i> | MQAT_NO_CONTEXT | 0 |
| <i>PutApplName</i> | None | 空字符串或空白 |
| <i>PutDate</i> | None | 空字符串或空白 |
| <i>PutTime</i> | None | 空字符串或空白 |
| <i>ApplOriginData</i> | None | 空字符串或空白 |
| <i>GroupId</i> | MQGI_NONE | Null |
| <i>MsgSeqNumber</i> | None | 1 |
| <i>Offset</i> | None | 0 |
| <i>MsgFlags</i> | MQMF_NONE | 0 |
| <i>OriginalLength</i> | MQOL_UNDEFINED | -1 |

注意:

1. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。
2. 在 C 编程语言中，宏变量 MQMD_DEFAULT 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值:

```
MQMD MyMD = {MQMD_DEFAULT};
```

C 声明

```
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 声明

```

** 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.

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

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_CODEDCCHARSETID 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)

```

Visual Basic 声明

```

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-消息描述符扩展

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|-----------------------|------------------|--------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>StrucLength</i> | MQMDE 结构的长度 | StrucLength |
| <i>Encoding</i> | MQMDE 之后数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | MQMDE 之后数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | MQMDE 之后数据的格式名称 | FORMAT |
| <i>Flags</i> | 常规标志 | 标志 |

| 字段 | 描述 | Topic |
|-----------------------|-------------------|--------------------------------|
| <i>GroupId</i> | 组标识 | GroupId |
| <i>MsgSeqNumber</i> | 组内逻辑消息的序列号 | MsgSeqNumber |
| <i>Offset</i> | 从逻辑消息开始的物理消息数据偏移量 | 偏移量 |
| <i>MsgFlags</i> | 消息标志 | MsgFlags |
| <i>OriginalLength</i> | 原始消息的长度 | OriginalLength |

MQMDE 概述

可用性: 所有 WebSphere MQ 系统, 以及连接到这些系统的 WebSphere MQ 客户机。

用途: MQMDE 结构描述有时发生在应用程序消息数据之前的数据。该结构包含 version-2 MQMD 中存在但在 version-1 MQMD 中不存在的那些 MQMD 字段。

格式名: MQFMT_MD_EXTENSION。

字符集和编码: MQMDE 中的数据必须采用本地队列管理器的字符集和编码; 这些数据由 C 编程语言的 *CodedCharSetId* queue-manager 属性和 MQENC_NATIVE 提供。

将 MQMDE 的字符集和编码设置到 *CodedCharSetId* 和 *Encoding* 字段中:

- MQMD (如果 MQMDE 结构位于消息数据的开头), 或者
- MQMDE 结构之前的头结构 (所有其他情况)。

如果 MQMDE 不在队列管理器的字符集和编码中, 那么将接受但不接受 MQMDE, 即将 MQMDE 视为消息数据。

注: 在 Windows 上, 使用 Micro Focus COBOL 编译的应用程序使用与队列管理器编码不同的 MQENC_NATIVE 值。虽然 MQPUT, MQPUT1 和 MQGET 调用上的 MQMD 结构中的数字字段必须采用 Micro Focus COBOL 编码, 但 MQMDE 结构中的数字字段必须采用队列管理器的编码。后者由 MQENC_NATIVE 为 C 编程语言提供, 并且具有值 546。

用法: 使用 version-2 MQMD 的应用程序不会迁到 MQMDE 结构。但是, 专用应用程序以及继续使用 version-1 MQMD 的应用程序在某些情况下可能会迁到 MQMDE。MQMDE 结构可能在以下情况下发生:

- 在 MQPUT 和 MQPUT1 调用上指定
- MQGET 调用返回
- 在传输队列上的消息中

MQPUT 和 MQPUT1 调用: 在 MQPUT 和 MQPUT1 调用上, 如果应用程序提供 version-1 MQMD, 那么应用程序可以选择使用 MQMDE 作为消息数据的前缀, 将 MQMD 中的 *Format* 字段设置为 MQFMT_MD_EXTENSION 以指示存在 MQMDE。如果应用程序未提供 MQMDE, 那么队列管理器将为 MQMDE 中的字段采用缺省值。队列管理器使用的缺省值与结构的初始值相同; 请参阅第 410 页的表 518。

如果应用程序提供了 version-2 MQMD, 并且以 MQMDE 作为应用程序消息数据的前缀, 那么将处理结构, 如第 407 页的表 517 中所示。

| MQMD 版本 | version-2 字段的值 | MQMDE 中相应字段的值 | 队列管理器执行的操作 |
|---------|----------------|---------------|-----------------|
| 1 | - | 有效 | 已采用 MQMDE |
| 2 | 缺省 | 有效 | 已采用 MQMDE |
| 2 | 不是缺省值 | 有效 | MQMDE 被视为消息数据 |
| 1 或 2 个 | 任何 | 无效 | 调用失败, 并带有相应的原因码 |

表 517: 针对 MQMDE 在 MQPUT 或 MQPUT1 上指定 MQMDE 时的队列管理器操作 (继续)

| MQMD 版本 | version-2 字段的值 | MQMDE 中相应字段的值 | 队列管理器执行的操作 |
|---------|----------------|-----------------------------|---------------|
| 1 或 2 个 | 任何 | MQMDE 的字符集或编码错误, 或者是不受支持的版本 | MQMDE 被视为消息数据 |

注: 在 z/OS 上, 如果应用程序指定了带有 MQMDE 的 version-1 MQMD, 那么仅当队列的 *IndexType* 为 MQIT_GROUP_ID 时, 队列管理器才会验证 MQMDE。

有一个特殊情况。如果应用程序使用 version-2 MQMD 来放置作为段的消息 (即, 设置了 MQMF_SEGMENT 或 MQMF_LAST_SEGMENT 标志), 并且 MQMD 中的格式名称为 MQFMT_DEAD_LETTER_HEADER, 那么队列管理器会生成 MQMDE 结构, 并将其插入到 MQDLH 结构及其后的数据之间。在队列管理器随消息保留的 MQMD 中, version-2 字段设置为其缺省值。

version-2 MQMD 中存在但不存在 version-1 MQMD 的多个字段是 MQPUT 和 MQPUT1 上的输入/输出字段。但是, 在 MQPUT 和 MQPUT1 调用的输出中, 队列管理器不会返回 MQMDE 中等效字段中的任何值; 如果应用程序需要这些输出值, 那么它必须使用 version-2 MQMD。

MQGET 调用返回的 MQMDE: 在 MQGET 调用上, 如果应用程序提供了 version-1 MQMD, 那么队列管理器会将 MQMDE 返回的消息作为前缀, 但前提是 MQMDE 中的一个或多个字段具有非缺省值。队列管理器将 MQMD 中的 *Format* 字段设置为值 MQFMT_MD_EXTENSION, 以指示存在 MQMDE。

如果应用程序在 *Buffer* 参数启动时提供 MQMDE, 那么将忽略 MQMDE。从 MQGET 调用返回时, 会将其替换为消息的 MQMDE (如果需要) 或由应用程序消息数据覆盖 (如果不需要 MQMDE)。

如果 MQGET 调用返回 MQMDE, 那么 MQMDE 中的数据通常采用队列管理器的字符集和编码。但是, 在下列情况下, MQMDE 可能使用其他某个字符集和编码:

- MQMDE 被视为 MQPUT 或 MQPUT1 调用上的数据 (请参阅第 407 页的表 517 以了解可能导致此情况的情况)。
- 从通过 TCP 连接连接的远程队列管理器接收到消息, 并且未正确设置接收消息通道代理程序 (MCA)。

注: 在 Windows 上, 使用 Micro Focus COBOL 编译的应用程序使用不同于队列管理器编码的 MQENC_NATIVE 值 (请参阅上文)。

传输队列上的消息中的 MQMDE: 传输队列上的消息以 MQXQH 结构为前缀, 其中包含 version-1 MQMD。MQMDE 也可能存在, 位于 MQXQH 结构和应用程序消息数据之间, 但通常仅当 MQMDE 中的一个或多个字段具有非缺省值时才存在。

其他 MQ 头结构也可以出现在 MQXQH 结构与应用程序消息数据之间。例如, 当死信头 MQDLH 存在且消息不是段时, 顺序为:

- MQXQH (包含 version-1 MQMD)
- MQMDE
- MQDLH
- 应用程序消息数据

MQMDE 的字段

MQMDE 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

CodedCharSetId (MQLONG)

这指定 MQMDE 结构之后的数据的字符集标识; 它不适用于 MQMDE 结构本身中的字符数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。队列管理器不会检查此字段是否有效。可以使用以下特殊值:

MQCCSI_INHERIT

遵循此结构的数据中的字符数据与此结构位于同一字符集中。

队列管理器将消息中发送的结构中的此值更改为结构的实际字符集标识。如果未发生错误，那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER，那么无法使用 MQCCSI_INHERIT。

此值在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

此字段的初始值为 MQCCSI_UNDEFINED。

Encoding (MQLONG)

这指定 MQMDE 结构之后的数据的数字编码; 它不适用于 MQMDE 结构本身中的数字数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。队列管理器不会检查该字段是否有效。请参阅第 364 页的『MQMD - 消息描述符』中描述的 *Encoding* 字段, 以获取有关数据编码的更多信息。

此字段的初始值为 MQENC_NATIVE。

Flags (MQLONG)

可以指定以下标志:

MQMDEF_NONE

没有标志。

此字段的初始值为 MQMDEF_NONE。

Format (MQCHAR8)

这指定遵循 MQMDE 结构的数据的格式名称。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。队列管理器不会检查此字段是否有效。请参阅第 364 页的『MQMD - 消息描述符』中描述的 *Format* 字段, 以获取有关格式名称的更多信息。

此字段的初始值为 MQFMT_NONE。

GroupId (MQBYTE24)

请参阅第 364 页的『MQMD - 消息描述符』中描述的 *GroupId* 字段。此字段的初始值为 MQGI_NONE。

MsgFlags (MQLONG)

请参阅第 364 页的『MQMD - 消息描述符』中描述的 *MsgFlags* 字段。此字段的初始值为 MQMF_NONE。

MsgSeqNumber (MQLONG)

请参阅第 364 页的『MQMD - 消息描述符』中描述的 *MsgSeqNumber* 字段。此字段的初始值为 1。

Offset (MQLONG)

请参阅第 364 页的『MQMD - 消息描述符』中描述的 *Offset* 字段。此字段的初始值为 0。

OriginalLength (MQLONG)

请参阅第 364 页的『MQMD - 消息描述符』中描述的 *OriginalLength* 字段。此字段的初始值为 MQOL_UNDEFINED。

StrucId (MQCHAR4)

该值必须为:

MQMDE_STRUC_ID

消息描述符扩展结构的标识。

对于 C 编程语言, 还定义了常量 MQMDE_STRUC_ID_ARRAY; 此值与 MQMDE_STRUC_ID 相同, 但是字符数组而不是字符串。

此字段的初始值为 MQMDE_STRUC_ID。

StrucLength (MQLONG)

这是 MQMDE 结构的长度; 定义了以下值:

MQMDE_LENGTH_2

version-2 消息描述符扩展结构的长度。

此字段的初始值为 MQMDE_LENGTH_2。

Version (MQLONG)

这是结构版本号; 值必须为:

MQMDE_VERSION_2

Version-2 消息描述符扩展结构。

以下常量指定当前版本的版本号:

MQMDE_CURRENT_VERSION

消息描述符扩展结构的当前版本。

此字段的初始值为 MQMDE_VERSION_2。

MQMDE 的初始值和语言声明

| 表 518: MQMDE 的 MQMDE 中字段的初始值 | | |
|------------------------------|------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQMDE_STRUC_ID | 'MDE↵' |
| <i>Version</i> | MQMDE_VERSION_2 | 2 |
| <i>StrucLength</i> | MQMDE_LENGTH_2 | 72 |
| <i>Encoding</i> | MQENC_NATIVE | 取决于环境 |
| <i>CodedCharSetId</i> | MQCCSI_UNDEFINED | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQMDEF_NONE | 0 |
| <i>GroupId</i> | MQGI_NONE | Null |
| <i>MsgSeqNumber</i> | None | 1 |
| <i>Offset</i> | None | 0 |
| <i>MsgFlags</i> | MQMF_NONE | 0 |
| <i>OriginalLength</i> | MQOL_UNDEFINED | -1 |

注意:

1. 符号 ↵ 表示单个空白字符。
2. 在 C 编程语言中, 宏变量 MQMDE_DEFAULT 包含以上列出的值。 它可以通过以下方式用于为结构中的字段提供初始值:

```
MQMDE MyMDE = {MQMDE_DEFAULT};
```

C 声明

```
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 声明

```

** 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-STRUCLNGTH 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.

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

MQMDE          DSECT

```

```

MQMDE_STRUCID      DS    CL4   Structure identifier
MQMDE_VERSION      DS    F     Structure version number
MQMDE_STRUCLNGTH   DS    F     Length of MQMDE structure
MQMDE_ENCODING     DS    F     Numeric encoding of data that follows
*
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_MSGSEQUNUMBER 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)

```

Visual Basic 声明

```

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'
  CodedCharSetId As Long   'MQMDE'
  CodedCharSetId As Long   'Character-set identifier of data that'
  Format       As String*8 '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'
  Offset       As Long     'group'
  Offset       As Long     'Offset of data in physical message from'
  MsgFlags     As Long     'start of logical message'
  OriginalLength As Long   'Message flags'
  OriginalLength As Long   'Length of original message'
End Type

```

MQMHBO-消息句柄到缓冲区选项

下表汇总了结构中的字段。MQMHBO 结构-缓冲区选项的消息句柄

| 表 519: MQMHBO 中的字段 | | |
|--------------------|------------------|-------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 控制 MQMHBUF 操作的选项 | 选项 |

MQMHBO 概述

可用性: 所有 WebSphere MQ 系统和 WebSphere MQ MQI 客户机。

用途:MQMHBO 结构允许应用程序指定用于控制如何从消息句柄生成缓冲区的选项。此结构是 MQMHBUF 调用上的输入参数。

字符集和编码:MQMHBO 中的数据必须在应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQMHBO 的字段

消息句柄到缓冲区选项结构-字段

MQMHBO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

选项 (MQLONG)

消息句柄到缓冲区选项结构-"选项" 字段

这些选项控制 MQMHBUF 的操作。

必须指定以下选项:

MQMHBO_PROPERTIES_IN_MQRFH2

将属性从消息句柄转换为缓冲区时, 请将其转换为 MQRFH2 格式。

(可选) 您还可以指定以下值。如果需要的值可以是:

- 一起添加 (请勿多次添加同一常量), 或者
- 通过按位“或”运算组合在一起 (如果编程语言支持位运算)。

MQMHBO_DELETE_PROPERTIES

将从消息句柄中删除添加到缓冲区的属性。如果调用失败, 那么不会删除任何属性。

这始终是一个输入字段。此字段的初始值为 MQMHBO_PROPERTIES_IN_MQRFH2。

StrucId (MQCHAR4)

消息句柄到缓冲区选项结构- StrucId 字段

这是结构标识。该值必须为:

MQMHBO_STRUC_ID

消息句柄到缓冲区选项结构的标识。

对于 C 编程语言, 还定义了常量 MQMHBO_STRUC_ID_ARRAY; 此值与 MQMHBO_STRUC_ID 相同, 但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQMHBO_STRUC_ID。

Version (MQLONG)

消息句柄到缓冲区选项结构-"版本" 字段

这是结构版本号。该值必须为:

MQMHBO_VERSION_1

消息句柄到缓冲区选项结构的版本号。

以下常量指定当前版本的版本号:

MQMHBO_CURRENT_VERSION

当前版本的消息句柄到缓冲区选项结构。

这始终是一个输入字段。此字段的初始值为 MQMHBO_VERSION_1。

MQMHBO 的初始值和语言声明

消息句柄到缓冲区结构-初始值

| 字段名称 | 常量的名称 | 常量值 |
|----------------|-----------------------------|--------|
| <i>StrucId</i> | MQMHBO_STRUC_ID | 'MHBO' |
| <i>Version</i> | MQMHBO_VERSION_1 | 1 |
| <i>Options</i> | MQMHBO_PROPERTIES_IN_MQRFH2 | |

表 520: MQMHBO 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|---|-------|-----|
| 注意: | | |
| 1. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。 2. 在 C 编程语言中，宏变量 MQMHBO_DEFAULT 包含上面列出的值。通过以下方式使用它来为结构中的字段提供初始值： | | |
| <pre>MQMHBO MyMHBO = {MQMHBO_DEFAULT};</pre> | | |

C 声明

消息句柄到缓冲区选项结构-C 语言声明

```
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 声明

缓冲区选项结构的消息句柄-COBOL 语言声明

```
** 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.
```

PL/I 声明

缓冲区选项结构的消息句柄-PL/I 语言声明

```
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 */
```

High Level Assembler 声明

消息句柄到缓冲区选项结构-汇编程序语言声明

```
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-对象描述符

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|--|-------------------------|-------------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>ObjectType</i> | 对象类型 | ObjectType |
| <i>ObjectName</i> | 对象名 | ObjectName |
| <i>ObjectQMgrName</i> | 对象队列管理器名称 | ObjectQMgrName |
| <i>DynamicQName</i> | 动态队列名称 | DynamicQName |
| <i>AlternateUserId</i> | 备用用户标识 | AlternateUserId |
| 注: 如果 <i>Version</i> 小于 MQOD_VERSION_2, 那么将忽略其余字段。 | | |
| <i>RecsPresent</i> | 存在的对象记录数 | RecsPresent |
| <i>KnownDestCount</i> | 成功打开的本地队列数 | KnownDestCount |
| <i>UnknownDestCount</i> | 成功打开的远程队列数 | UnknownDestCount |
| <i>InvalidDestCount</i> | 未能打开的队列数 | InvalidDestCount |
| <i>ObjectRecOffset</i> | 第一个对象记录与 MQOD 开始处之间的偏移量 | ObjectRec 偏移量 |
| <i>ResponseRecOffset</i> | 第一个响应记录与 MQOD 开始处之间的偏移量 | ResponseRecOffset |
| <i>ObjectRecPtr</i> | 第一个对象记录的地址 | ObjectRecPtr |
| <i>ResponseRecPtr</i> | 第一个响应记录的地址 | ResponseRecPtr |
| 注: 如果 <i>Version</i> 小于 MQOD_VERSION_3, 那么将忽略其余字段。 | | |
| <i>AlternateSecurityId</i> | 备用安全标识 | AlternateSecurityId |
| <i>ResolvedQName</i> | 已解析队列名称 | ResolvedQName |
| <i>ResolvedQMgrName</i> | 已解析队列管理器名称 | ResolvedQMgrName |
| 注: 如果 <i>Version</i> 小于 MQOD_VERSION_4, 那么将忽略其余字段。 | | |
| <i>ObjectString</i> | 长对象名 | ObjectString |
| <i>SelectionString</i> | 选择字符串 | SelectionString |
| <i>ResObjectString</i> | 已解析的长对象名 | ResObject 字符串 |
| <i>ResolvedType</i> | 已解析的对象类型 | ResolvedType |

MQOD 概述

可用性: 所有 WebSphere MQ 系统, 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: MQOD 结构用于按名称指定对象。以下类型的对象有效:

- 队列或分发列表
- 名称列表
- 进程定义
- 队列管理器
- Topic

此结构是 MQOPEN 和 MQPUT1 调用上的输入/输出参数。

版本:MQOD 的当前版本为 MQOD_VERSION_4。要在多个环境之间移植的应用程序必须确保在所有相关环境中支持所需的 MQOD 版本。仅在结构的最新版本中存在的字段在随后的描述中标识为此类字段。

为受支持的编程语言提供的头 COPY 和 INCLUDE 文件包含环境支持的最新版本的 MQOD，但 *Version* 字段的初始值设置为 MQOD_VERSION_1。要使用 version-1 结构中不存在的字段，应用程序必须将 *Version* 字段设置为所需版本的版本号。

要打开分发列表，*Version* 必须为 MQOD_VERSION_2 或更高版本。

字符集和编码:MQOD 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是，如果应用程序作为 MQ MQI 客户机运行，那么该结构必须采用客户机的字符集和编码。

MQOD 的字段

MQOD 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

AlternateSecurity 标识 (MQBYTE40)

这是随 *AlternateUserId* 一起传递到授权服务的安全标识，以允许执行相应的授权检查。仅当以下情况下才使用 *AlternateSecurityId*:

- MQOO_ALTERNATE_USER_AUTHORITY 是在 MQOPEN 调用上指定的，或者
- 在 MQPUT1 调用上指定了 MQPMO_ALTERNATE_USER_AUTHORITY，

和 *AlternateUserId* 字段并非完全空白，直到第一个空字符或字段结束。

在 Windows 上，*AlternateSecurityId* 可用于提供唯一标识 *AlternateUserId* 的 Windows 安全标识 (SID)。可以使用 *LookupAccountName()* Windows API 调用从 Windows 系统获取用户的 SID。

在 z/OS 上，将忽略此字段。

AlternateSecurityId 字段具有以下结构:

- 第一个字节是一个二进制整数，其中包含后续重要数据的长度; 该值不包括长度字节本身。如果不存在安全标识，那么长度为零。
- 第二个字节指示存在的安全标识的类型; 可以使用以下值:

MQSIDT_NT_SECURITY_ID

Windows 安全标识。

MQSIDT_NONE

无安全标识。

- 由第一个字节定义的最大长度的第三个字节和后续字节包含安全标识本身。
- 字段中的剩余字节设置为二进制零。

可以使用以下特殊值:

MQSID_NONE

未指定安全标识。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQSID_NONE_ARRAY; 此值与 MQSID_NONE 相同，但是字符数组而不是字符串。

这是一个输入字段。此字段的长度由 MQ_SECURITY_ID_LENGTH 指定。此字段的初始值为 MQSID_NONE。如果 *Version* 小于 MQOD_VERSION_3，那么将忽略此字段。

AlternateUser 标识 (MQCHAR12)

如果对 MQOPEN 调用指定 MQOO_ALTERNATE_USER_AUTHORITY，或对 MQPUT1 调用指定 MQPMO_ALTERNATE_USER_AUTHORITY，那么此字段包含用于检查打开权限的备用用户标识，以代替当前运行应用程序的用户标识。但是，仍使用当前用户标识执行某些检查 (例如，上下文检查)。

如果指定了 MQOO_ALTERNATE_USER_AUTHORITY 或 MQPMO_ALTERNATE_USER_AUTHORITY，并且此字段完全为空白，直到第一个空字符或字段末尾，那么仅当不需要用户授权即可使用指定的选项打开此对象时，打开才能成功。

如果既未指定 MQOO_ALTERNATE_USER_AUTHORITY，也未指定 MQPMO_ALTERNATE_USER_AUTHORITY，那么将忽略此字段。

在所指示的环境中存在以下差异：

- 在 z/OS 上，仅使用 *AlternateUserId* 的前 8 个字符来检查打开的权限。但是，必须授权当前用户标识指定此特定备用用户标识；备用用户标识的所有 12 个字符都用于此检查。用户标识必须仅包含外部安全管理器允许的字符。

如果为队列指定了 *AlternateUserId*，那么在放入消息时，队列管理器可以随后使用该值。如果在 MQPUT 或 MQPUT1 调用上指定的 MQPMO_*_CONTEXT 选项导致队列管理器生成身份上下文信息，那么队列管理器会将 *AlternateUserId* 放入消息的 MQMD 中的 *UserIdentifier* 字段中，以代替当前用户标识。

- 在其他环境中，*AlternateUserId* 仅用于对要打开的对象进行访问控制检查。如果对象是队列，那么 *AlternateUserId* 不会影响使用该队列句柄发送的消息的 MQMD 中 *UserIdentifier* 字段的内容。

这是一个输入字段。此字段的长度由 MQ_USER_ID_LENGTH 给出。此字段的初始值是 C 中的空字符串，以及其他编程语言中的 12 个空白字符。

DynamicQName (MQCHAR48)

这是由 MQOPEN 调用创建的动态队列的名称。仅当 *ObjectName* 指定模型队列的名称时，这才具有相关性；在所有其他情况下，将忽略 *DynamicQName*。

名称中有效的字符与 *ObjectName* 中有效的字符相同，但星号也有效。如果 *ObjectName* 是模型队列的名称，那么空白的名称（或者在第一个空字符之前仅出现空白的名称）无效。

如果名称中的最后一个非空白字符是星号（*），那么队列管理器会将星号替换为字符串，以保证为队列生成的名称在本地队列管理器上是唯一的。为了允许有足够数量的字符用于此目的，星号仅在位置 1 到 33 中有效。不得有空格以外的字符或星号后面的空字符。

星号出现在第一个字符位置是有效的，在这种情况下，名称仅由队列管理器生成的字符组成。

在 z/OS 上，请勿在第一个字符位置中使用带有星号的名称，因为不会对具有自动生成的全名的队列进行安全性检查。

这是一个输入字段。此字段的长度由 MQ_Q_NAME_LENGTH 提供。此字段的初始值由环境确定：

- 在 z/OS 上，值为 'CSQ.*'。
- 在其他平台上，值为 'AMQ.*'。

该值是 C 中的以 null 结束的字符串，以及其他编程语言中的空白填充字符串。

InvalidDest 计数 (MQLONG)

这是分发列表中未能成功打开的队列数。如果存在，那么在打开不在分发列表中的单个队列时也会设置此字段。

注：如果存在此字段，那么仅在 MQOPEN 或 MQPUT1 调用上的 *CompCode* 参数为 MQCC_OK 或 MQCC_WARNN；如果 *CompCode* 参数为 MQCC_FAILED，那么未设置此字段。

这是输出字段。此字段的初始值为 0。如果 *Version* 小于 MQOD_VERSION_1，那么将忽略此字段。

KnownDest 计数 (MQLONG)

这是分发列表中解析为本地队列并成功打开的队列数。此计数不包括解析为远程队列的队列（即使最初使用本地传输队列来存储消息）。如果存在，那么在打开不在分发列表中的单个队列时也会设置此字段。

这是输出字段。此字段的初始值为 0。如果 *Version* 小于 MQOD_VERSION_1，那么将忽略此字段。

ObjectName (MQCHAR48)

这是由 *ObjectQMgrName* 标识的队列管理器上定义的对象局部名。此名称可包含以下字符：

- 大写字母字符 (A 到 Z)
- 小写字母字符 (a 到 z)
- 数字位 (0 到 9)
- 句点 (.)、正斜杠 (/)、下划线 (_)、百分号 (%)

此名称不得包含前导空格或嵌入空格，但可以包含尾部空格。使用空字符来指示名称中重要数据的结束；空字符及其后的任何字符被视为空白。以下限制适用于指示的环境中：

- 在使用 EBCDIC 片假名的系统上，不得使用小写字母。
- 在 z/OS 上：
 - 避免使用以下划线开头或结尾的名称；操作和控制面板无法处理这些名称。
 - 百分号字符对于 RACF 具有特殊含义。如果使用 RACF 作为外部安全性管理器，那么名称不得包含百分比。如果这样做，那么在使用 RACF 通用概要文件时，这些名称不会包含在任何安全性检查中。
- 在 IBM i 上，当在命令上指定时，必须将包含小写字母，正斜杠或百分号的名称括在引号中。对于在结构中作为字段出现的名称或在调用时作为参数出现的名称，不得指定这些引号。

可以从两个不同的字段构建完整主题名称：*ObjectName* 和 *ObjectString*。有关如何使用这两个字段的详细信息，请参阅第 504 页的『使用主题字符串』。

以下要点适用于所指示对象的类型：

- 如果 *ObjectName* 是模型队列的名称，那么队列管理器将使用模型队列的属性创建动态队列，并在 *ObjectName* 字段中返回所创建队列的名称。只能在 MQOPEN 调用上指定模型队列；模型队列在 MQPUT1 调用上无效。
- 如果 *ObjectName* 是具有 TARGTYPE (TOPIC) 的别名队列的名称，那么将首先对指定的别名队列进行安全性检查；当使用别名队列时，这是正常情况。当安全性检查成功完成时，MQOPEN 调用将继续，并将像 MQOT_TOPIC 上的 MQOPEN 调用一样运行；这包括对管理主题对象进行安全性检查。
- 如果 *ObjectName* 和 *ObjectQMgrName* 标识本地队列管理器所属的队列共享组所拥有的共享队列，那么本地队列管理器上也不得存在同名的队列定义。如果存在这样的定义 (本地队列，别名队列，远程队列或模型队列)，那么调用将失败，原因码为 MQRC_OBJECT_NOT_UNIQUE。
- 如果要打开的对象是分发列表 (即，*RecsPresent* 存在且大于零)，那么 *ObjectName* 必须为空或为空字符串。如果不满足此条件，那么调用将失败，原因码为 MQRC_OBJECT_NAME_ERROR。
- 如果 *ObjectType* 是 MQOT_Q_MGR，那么将应用特殊规则；在这种情况下，名称必须完全为空白，直到第一个空字符或字段结束。

当 *ObjectName* 是模型队列的名称时，这是 MQOPEN 调用的输入/输出字段，在所有其他情况下，这是仅输入字段。此字段的长度由 MQ_Q_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ObjectQMgr 名称 (MQCHAR48)

这是定义了 *ObjectName* 对象的队列管理器的名称。名称中有效的字符与 *ObjectName* 的字符相同 (请参阅第 417 页的『*ObjectName* (MQCHAR48)』)。名称完全为空白，直到第一个空字符或字段的末尾表示应用程序所连接的队列管理器 (本地队列管理器)。

以下要点适用于所指示对象的类型：

- 如果 *ObjectType* 是 MQOT_TOPIC，MQOT_NAMELIST，MQOT_PROCESS 或 MQOT_Q_MGR，那么 *ObjectQMgrName* 必须为空白或本地队列管理器的名称。
- 如果 *ObjectName* 是模型队列的名称，那么队列管理器将使用模型队列的属性创建动态队列，并在 *ObjectQMgrName* 字段中返回创建队列的队列管理器的名称；这是本地队列管理器的名称。只能在 MQOPEN 调用上指定模型队列；模型队列在 MQPUT1 调用上无效。
- 如果 *ObjectName* 是集群队列的名称，并且 *ObjectQMgrName* 为空白，那么使用 MQOPEN 调用返回的队列句柄发送的消息的目标由队列管理器 (或集群工作负载出口，如果已安装) 选择，如下所示：
 - 如果指定了 MQOO_BIND_ON_OPEN，那么队列管理器将在处理 MQOPEN 调用时选择集群队列的特定实例，并且使用此队列句柄放入的所有消息都将发送到该实例。

- 如果指定了 MQOO_BIND_NOT_FIXED，那么对于使用此队列句柄的每个连续 MQPUT 调用，队列管理器可以选择目标队列的不同实例 (驻留在集群中的不同队列管理器上)。

如果应用程序需要将消息发送到集群队列的特定实例 (即，位于集群中特定队列管理器上的队列实例)，那么应用程序必须在 *ObjectQMGrName* 字段中指定该队列管理器的名称。这将强制本地队列管理器将消息发送到指定的目标队列管理器。

- 如果 *ObjectName* 是共享队列的名称，由本地队列管理器所属的队列共享组拥有，*ObjectQMGrName* 可以是队列共享组的名称，本地队列管理器的名称或空白；消息放在同一队列上，无论这些值中的任何一个已指定。

仅在 z/OS 上支持队列共享组。

- 如果 *ObjectName* 是由远程队列共享组 (即，本地队列管理器不属于的队列共享组) 拥有的共享队列的名称，那么 *ObjectQMGrName* 必须是队列共享组的名称。您可以使用属于该组的队列管理器的名称，但如果该特定队列管理器在消息到达队列共享组时不可用，那么这可能会延迟消息。
- 如果要打开的对象是分发列表 (即，*RecsPresent* 大于零)，那么 *ObjectQMGrName* 必须为空白或空字符串。如果不满足此条件，那么调用将失败，原因码为 MQRC_OBJECT_Q_MGR_NAME_ERROR。

当 *ObjectName* 是模型队列的名称时，这是 MQOPEN 调用的输入/输出字段，在所有其他情况下，这是仅输入字段。此字段的长度由 MQ_Q_MGR_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ObjectRec 偏移量 (MQLONG)

这是从 MQOD 结构开始的第一个 MQOR 对象记录的偏移量 (以字节为单位)。偏移可以是正数或负数。仅当正在打开分发列表时，才会使用 *ObjectRecOffset*。如果 *RecsPresent* 为零，那么将忽略该字段。

打开分发列表时，必须提供一个或多个 MQOR 对象记录的数组，以便在分发列表中指定目标队列的名称。可以通过以下两种方法之一来完成此操作：

- 通过使用偏移量字段 *ObjectRecOffset*。

在这种情况下，应用程序必须声明自己的结构，该结构包含后跟 MQOR 记录数组的 MQOD (具有所需数量的数组元素)，并将 *ObjectRecOffset* 设置为该数组中第一个元素从 MQOD 开始的偏移量。请确保此偏移量是正确的，并且具有可在 MQLONG 中容纳的值 (最严格的编程语言是 COBOL，其有效范围为 -999 999 999 到 +999 999 999 999)。

对于不支持指针数据类型或以无法移植到不同环境 (例如，COBOL 编程语言) 的方式实现指针数据类型的编程语言，请使用 *ObjectRecOffset*。

- 通过使用指针字段 *ObjectRecPtr*。

在此情况下，应用程序可独立于 MQOD 结构声明 MQOR 结构的数组，并将 *ObjectRecPtr* 设置为数组的地址。

将 *ObjectRecPtr* 用于以可移植到不同环境 (例如 C 编程语言) 的方式支持指针数据类型的编程语言。

无论您选择何种技术，请使用 *ObjectRecOffset* 和 *ObjectRecPtr* 之一；如果两者都为零或都为非零，那么调用将失败，原因码为 MQRC_OBJECT_RECORDS_ERROR。

这是一个输入字段。此字段的初始值为 0。如果 *Version* 小于 MQOD_VERSION_2，那么将忽略此字段。

ObjectRecPtr (MQPTR)

这是第一个 MQOR 对象记录的地址。仅当正在打开分发列表时，才会使用 *ObjectRecPtr*。如果 *RecsPresent* 为零，那么将忽略该字段。

您可以使用 *ObjectRecPtr* 或 *ObjectRecOffset* 来指定对象记录，但不能同时指定这两个对象记录；请参阅上面 *ObjectRecOffset* 字段的描述以获取详细信息。如果不使用 *ObjectRecPtr*，请将其设置为空指针或空字节。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针，否则为全空字节字符串。如果 *Version* 小于 MQOD_VERSION_2，那么将忽略此字段。

注：在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串，初始值为全空字节字符串。

ObjectString (MQCHARV)

ObjectString 字段指定长对象名。

这指定要使用的长对象名。仅针对 *ObjectType* 的某些值引用此字段，而针对所有其他值忽略此字段。请参阅 *ObjectType* 的描述，以获取指示使用此字段的值的详细信息。

如果未正确指定 *ObjectString*，那么根据如何使用 MQCHARV 结构的描述，或者如果该结构超过最大长度，那么调用将失败，原因码为 MQRC_OBJECT_STRING_ERROR。

这是一个输入字段。此结构中字段的初始值与 MQCHARV 结构中的初始值相同。

可以从两个不同的字段构建完整主题名称: *ObjectName* 和 *ObjectString*。有关如何使用这两个字段的详细信息，请参阅第 504 页的『使用主题字符串』。

ObjectType (MQLONG)

要在对象描述符中命名的对象的类型。可能的值为:

MQOT_CLNTCONN_CHANNEL

客户机连接通道。在 *ObjectName* 字段中找到对象的名称。

MQOT_Q

队列。在 *ObjectName* 字段中找到对象的名称。

MQOT_NAMELIST

NAMELIST. 在 *ObjectName* 字段中找到对象的名称

MQOT_PROCESS

process definition. 在 *ObjectName* 字段中找到对象的名称

MQOT_Q_MGR

队列管理器。在 *ObjectName* 字段中找到对象的名称

MQOT_TOPIC

主题中查看此版本新增功能的摘要。可以从两个不同的字段构建完整主题名称: *ObjectName* 和 *ObjectString*。

有关如何使用这两个字段的详细信息，请参阅第 504 页的『使用主题字符串』。

这始终是一个输入字段。此字段的初始值为 MQOT_Q。

RecsPresent (MQLONG)

这是应用程序提供的 MQOR 对象记录数。如果此数字大于零，那么表示正在打开分发列表，其中 *RecsPresent* 是列表中的目标队列数。分发列表只能包含一个目标。

RecsPresent 的值不得小于零，如果该值大于零，那么 *ObjectType* 必须为 MQOT_Q; 如果未满足这些条件，那么调用将失败，原因码为 MQRC_RECS_PRESENT_ERROR。

在 z/OS 上，此字段必须为零。

这是一个输入字段。此字段的初始值为 0。如果 *Version* 小于 MQOD_VERSION_2，那么将忽略此字段。

ResObject 字符串 (MQCHARV)

ResObject 字符串字段是队列管理器解析 *ObjectName* 字段中提供的名称后的长对象名称。

仅针对引用主题对象的主题和队列别名返回此字段。

如果 *ObjectString* 中提供了长对象名，而 *ObjectName* 中未提供任何内容，那么此字段中返回的值与 *ObjectString* 中提供的值相同。

如果省略此字段 (即 *ResObjectString.VSBufSize* 为零)，那么将不会返回 *ResObjectString*，但将在 *ResObjectString.VSLength* 中返回长度。

如果缓冲区长度 (在 *ResObjectString.VSBufSize* 中提供) 比完整的 *ResObjectString* 短，那么字符串将被截断，并将返回在提供的缓冲区中可容纳的最右边字符数。

如果未正确指定 *ResObjectString*，那么根据如何使用 MQCHARV 结构的描述，或者如果该结构超过最大长度，那么调用将失败，原因码为 MQRC_RES_OBJECT_STRING_ERROR。

ResolvedQMgr 名称 (MQCHAR48)

这是本地队列管理器解析名称后的目标队列管理器的名称。返回的名称是拥有由 *ResolvedQName* 标识的队列的队列管理器的名称。 *ResolvedQMgrName* 可以是本地队列管理器的名称。

如果 *ResolvedQName* 是本地队列管理器所属的队列共享组所拥有的共享队列，那么 *ResolvedQMgrName* 是队列共享组的名称。如果队列由其他某个队列共享组拥有，那么 *ResolvedQName* 可以是队列共享组的名称，也可以是作为队列共享组成员的队列管理器的名称 (返回的值的性质由本地队列管理器中存在的队列定义确定)。

仅当对象是打开用于浏览，输入或输出 (或任何组合) 的单个队列时，才会返回非空白值。如果打开的对象是下列任何一项，那么 *ResolvedQMgrName* 设置为空白：

- 不是队列
- 队列，但未打开以进行浏览，输入或输出
- 指定了 MQOO_BIND_NOT_FIXED 的集群队列 (或者当 *DefBind* 队列属性的值为 MQBND_BIND_NOT_FIXED 时生效的 MQOO_BIND_AS_Q_DEF)
- 分发列表

这是输出字段。此字段的长度由 MQ_Q_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。如果 *Version* 小于 MQOD_VERSION_3，那么将忽略此字段。

ResolvedQName (MQCHAR48)

这是本地队列管理器解析名称后的目标队列的名称。返回的名称是由 *ResolvedQMgrName* 标识的队列管理器上存在的队列的名称。

仅当对象是打开用于浏览，输入或输出 (或任何组合) 的单个队列时，才会返回非空白值。如果打开的对象是下列任何一项，那么 *ResolvedQName* 设置为空白：

- 不是队列
- 队列，但未打开以进行浏览，输入或输出
- 分发列表
- 引用主题对象的别名队列 (请改为参阅 [ResObjectString](#))。
- 解析为主题对象的别名队列。

这是输出字段。此字段的长度由 MQ_Q_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。如果 *Version* 小于 MQOD_VERSION_3，那么将忽略此字段。

ResolvedType (MQLONG)

要打开的已解析 (基本) 对象的类型。

可能的值为：

MQOT_Q

解析的对象是队列。当直接打开队列或打开指向队列的别名队列时，此值适用。

MQOT_TOPIC

已解析的对象是主题。当直接打开主题或打开指向主题对象的别名队列时，此值适用。

MQOT_NONE

解析的类型既不是队列，也不是主题。

ResponseRec 偏移量 (MQLONG)

这是从 MQOD 结构开始的第一个 MQRR 响应记录的偏移量 (以字节为单位)。偏移可以是正数或负数。仅当正在打开分发列表时，才会使用 *ResponseRecOffset*。如果 *RecsPresent* 为零，那么将忽略该字段。

打开分发列表时，可以提供一个或多个 MQRR 响应记录的数组，以标识未能打开的队列 (MQRR 中的 *CompCode* 字段) 以及每次失败的原因 (MQRR 中的 *Reason* 字段)。数据在响应记录数组中返回的顺序与队列名称在对象记录数组中出现的顺序相同。仅当调用结果混合时，队列管理器才会设置响应记录 (即，某些队列已成功打开，而其他队列失败，或者所有队列都失败，但原因不同)；来自调用的原因码 MQRC_MULTIPLE_REASON 指示此情况。如果相同原因码适用于所有队列，那么将在 MQOPEN 或

MQPUT1 调用的 *Reason* 参数中返回该原因，并且不会设置响应记录。响应记录是可选的，但如果提供了这些记录，那么其中必须有 *RecsPresent* 个记录。

可以采用与对象记录相同的方式提供响应记录，方法是在 *ResponseRecOffset* 中指定偏移量，或者在 *ResponseRecPtr* 中指定地址；有关如何执行此操作的详细信息，请参阅以上 *ObjectRecOffset* 的描述。但是，不能使用多个 *ResponseRecOffset* 和 *ResponseRecPtr*；如果两者都非零，那么调用将失败，原因码为 MQRC_RESPONSE_RECORDS_ERROR。

对于 MQPUT1 调用，这些响应记录用于返回有关将消息发送到分发列表中的队列时发生的错误以及打开队列时发生的错误的信息。仅当来自队列的完成代码为 MQCC_OK 或 MQCC_WARNING 时，来自队列的放置操作的完成代码和原因码才会替换来自该队列的打开操作的完成代码和原因码。

这是一个输入字段。此字段的初始值为 0。如果 *Version* 小于 MQOD_VERSION_2，那么将忽略此字段。

ResponseRecPtr (MQPTR)

这是第一个 MQRR 响应记录的地址。仅当正在打开分发列表时，才会使用 *ResponseRecPtr*。如果 *RecsPresent* 为零，那么将忽略该字段。

使用 *ResponseRecPtr* 或 *ResponseRecOffset* 指定响应记录，但不能同时指定这两者；请参阅上面 *ResponseRecOffset* 字段的描述以获取详细信息。如果不使用 *ResponseRecPtr*，请将其设置为空指针或空字节。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针，否则为全空字节字符串。如果 *Version* 小于 MQOD_VERSION_2，那么将忽略此字段。

注：在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串，初始值为全空字节字符串。

SelectionString (MQCHARV)

这是用于提供从队列中检索消息时使用的选择条件的字符串。

在以下情况下不得提供 *SelectionString*：

- 如果 *ObjectType* 不是 MQOT_Q
- 如果未使用 MQOO_BROWSE 或 MQOO_INPUT_* 选项之一打开正在打开的队列

如果在这些情况下提供了 *SelectionString*，那么调用将失败，原因码为 MQRC_SELECTOR_INVALID_FOR_TYPE。

如果未正确指定 *SelectionString*，那么根据如何使用第 263 页的『MQCHARV-变量长度字符串』结构的描述，或者如果该结构超过最大长度，那么调用将失败，原因码为 MQRC_SELECTION_STRING_ERROR。*SelectionString* 的最大长度为 MQ_SELECTOR_LENGTH。

SelectionString 用法在 [选择器](#) 中进行了描述。

StrucId (MQCHAR4)

这是结构标识；值必须为：

MQOD_STRUC_ID

对象描述符结构的标识。

对于 C 编程语言，还定义了常量 MQOD_STRUC_ID_ARRAY；此值与 MQOD_STRUC_ID 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQOD_STRUC_ID。

UnknownDest 计数 (MQLONG)

这是分发列表中解析为远程队列并成功打开的队列数。如果存在，那么在打开不在分发列表中的单个队列时也会设置此字段。

这是输出字段。此字段的初始值为 0。如果 *Version* 小于 MQOD_VERSION_1，那么将忽略此字段。

Version (MQLONG)

这是结构版本号; 值必须是下列其中一项:

MQOD_VERSION_1

Version-1 对象描述符结构。

MQOD_VERSION_2

Version-2 对象描述符结构。

MQOD_VERSION_3

Version-3 对象描述符结构。

MQOD_VERSION_4

Version-4 对象描述符结构。

所有版本在所有 WebSphere MQ V7.0 环境中都受支持。

仅在结构的最新版本中存在的字段在字段的描述中标识为此类字段。 以下常量指定当前版本的版本号:

MQOD_CURRENT_VERSION

对象描述符结构的当前版本。

这始终是一个输入字段。 此字段的初始值为 MQOD_VERSION_1。

MQOD 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|----------------------------|-----------------|-------------------------------|
| <i>StrucId</i> | MQOD_STRUC_ID | '0D-11' |
| <i>Version</i> | MQOD_VERSION_1 | 1 |
| <i>ObjectType</i> | MQOT_Q | 1 |
| <i>ObjectName</i> | None | 空字符串或空白 |
| <i>ObjectQMgrName</i> | None | 空字符串或空白 |
| <i>DynamicQName</i> | None | z/OS 上的 'CSQ.*' ; 否则为 'AMQ.*' |
| <i>AlternateUserId</i> | None | 空字符串或空白 |
| <i>RecsPresent</i> | None | 0 |
| <i>KnownDestCount</i> | None | 0 |
| <i>UnknownDestCount</i> | None | 0 |
| <i>InvalidDestCount</i> | None | 0 |
| <i>ObjectRecOffset</i> | None | 0 |
| <i>ResponseRecOffset</i> | None | 0 |
| <i>ObjectRecPtr</i> | None | 空指针或空字节 |
| <i>ResponseRecPtr</i> | None | 空指针或空字节 |
| <i>AlternateSecurityId</i> | MQSID_NONE | Null |
| <i>ResolvedQName</i> | None | 空字符串或空白 |
| <i>ResolvedQMgrName</i> | None | 空字符串或空白 |
| <i>ObjectString</i> | MQCHARV_DEFAULT | 为 MQCHARV 定义 |
| <i>SelectionString</i> | MQCHARV_DEFAULT | 为 MQCHARV 定义 |
| <i>ResObjectString</i> | MQCHARV_DEFAULT | 为 MQCHARV 定义 |
| <i>ResolvedType</i> | MQOT_NONE | 0 |

| 字段名称 | 常量的名称 | 常量值 |
|---|-------|-----|
| 注意: | | |
| 1. 符号 _ 表示单个空白字符。 | | |
| 2. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。 | | |
| 3. 在 C 编程语言中，宏变量 MQOD_DEFAULT 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值: | | |
| <pre>MQOD MyOD = {MQOD_DEFAULT};</pre> | | |

C 声明

```
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
    object type */
    /* Ver:4 */
};
```

COBOL 声明

```
** 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-OBJECTQMRNAME         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-KNOWNDSTCOUNT        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.

```

PL/I 声明

```

dcl
1 MQOD based,
3 StructId 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 */

```

High Level Assembler 声明

```

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_UNKNOWNDSTCOUNT 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_OBJECTRECPTTR  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_VSLLENGTH 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_VSLLENGTH 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_VSLLENGTH 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 |
| MQOD_AREA | DS | | CL(MQOD_LENGTH) |

Visual Basic 声明

```

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-对象记录

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|-----------------------|-----------|--------------------------------|
| <i>ObjectName</i> | 对象名 | ObjectName |
| <i>ObjectQMgrName</i> | 对象队列管理器名称 | ObjectQMgrName |

MQOR 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: 使用 MQOR 结构指定单个目标队列的队列名称和队列管理器名称。MQOR 是 MQOPEN 和 MQPUT1 调用的输入结构。

字符集和编码: MQOR 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集, 以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 MQ MQI 客户机运行, 那么该结构必须采用客户机的字符集和编码。

用法: 通过在 MQOPEN 调用上提供这些结构的数组, 可以打开队列列表; 此列表称为 分发列表。使用该 MQOPEN 调用返回的队列句柄放置的每条消息都放置在列表中的每个队列上, 前提是队列已成功打开。

MQOR 的字段

MQOR 结构包含以下字段; 这些字段按 **字母顺序**进行了描述:

ObjectName (MQCHAR48)

这与 MQOD 结构中的 *ObjectName* 字段相同 (请参阅 MQOD 以获取详细信息), 只是:

- 它必须是队列的名称。
- 它不能是模型队列的名称。

这始终是一个输入字段。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ObjectQMgr 名称 (MQCHAR48)

这与 MQOD 结构中的 *ObjectQMgrName* 字段相同 (请参阅 MQOD 以获取详细信息)。

这始终是一个输入字段。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

MQOR 的初始值和语言声明

| 表 522: MQOR 中用于 MQOR 的字段的初始值 | | |
|---|-------|---------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>ObjectName</i> | None | 空字符串或空白 |
| <i>ObjectQMgrName</i> | None | 空字符串或空白 |
| 注意: <ol style="list-style-type: none"> 1. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。 2. 在 C 编程语言中，宏变量 MQOR_DEFAULT 包含上面列出的值。它可以通过以下方式用于为结构中的字段提供初始值: <pre>MQOR MyOR = {MQOR_DEFAULT};</pre> | | |

C 声明

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
    MQCHAR48 ObjectName; /* Object name */
    MQCHAR48 ObjectQMgrName; /* Object queue manager name */
};
```

COBOL 声明

```
** MQOR structure
10 MQOR.
** Object name
15 MQOR-OBJECTNAME PIC X(48).
** Object queue manager name
15 MQOR-OBJECTQMGRNAME PIC X(48).
```

PL/I 声明

```
dcl
1 MQOR based,
3 ObjectName char(48); /* Object name */
3 ObjectQMgrName char(48); /* Object queue manager name */
```

Visual Basic 声明

```
Type MQOR
    ObjectName As String*48 'Object name'
    ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

MQPD-属性描述符

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|--------------------|------------|-----------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 选项 | 选项 |
| <i>Support</i> | 需要对消息属性的支持 | 支持 |
| <i>Context</i> | 属性所属的消息上下文 | 上下文 |
| <i>CopyOptions</i> | 复制属性所属的选项 | CopyOptions |

MQPD 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS 和 WebSphere MQ MQI 客户机。

用途: MQPD 用于定义属性的属性。此结构是 MQSETMP 调用上的输入/输出参数以及 MQINQMP 调用上的输出参数。

字符集和编码: MQPD 中的数据必须位于应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQPD 的字段

MQPD 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

上下文 (MQLONG)

这描述了该属性所属的消息上下文。

当队列管理器接收到包含由队列管理器识别为不正确的 WebSphere MQ 定义的属性的消息时, 队列管理器将更正 *Context* 字段的值。

可以指定以下选项:

MQPD_USER_CONTEXT

该属性与用户上下文相关联。

无需特殊授权即可使用 MQSETMP 调用来设置与用户上下文关联的属性。

在 WebSphere MQ V 7.0 队列管理器上, 将保存与用户上下文相关联的属性, 如 MQOO_SAVE_ALL_CONTEXT 所述。指定了 MQPMO_PASS_ALL_CONTEXT 的 MQPUT 调用会导致将属性从保存的上下文复制到新消息中。

如果不需要先前描述的选项, 那么可以使用以下选项:

MQPD_NO_CONTEXT

该属性未与消息上下文关联。

使用 *Reason* 代码 MQRC_PD_ERROR 拒绝无法识别的值

这是 MQSETMP 调用的输入/输出字段以及 MQINQMP 调用的输出字段。此字段的初始值为 MQPD_NO_CONTEXT。

CopyOptions (MQLONG)

这描述了应该将属性复制到的消息类型。这是已识别的 WebSphere MQ 定义的属性的仅输出字段; WebSphere MQ 设置相应的值。

当队列管理器接收到包含 WebSphere MQ 定义的属性 (队列管理器可识别为不正确) 的消息时, 队列管理器将更正 *CopyOptions* 字段的值。

您可以指定其中一个或多个选项, 如果需要多个选项, 那么值可以是:

- 相加（不要将同一常量相加多次），或者
- 通过按位“或”运算组合在一起（如果编程语言支持位运算）。

MQCOPY_FOR 何承天

此属性将复制到要转发的消息中。

MQCOPY_PUBLISH

当发布消息时，会将此属性复制到订户接收的消息中。

MQCOPY_REPLY

此属性将复制到应答消息中。

MQCOPY_REPORT

此属性将复制到报告消息中。

MQCOPY_ALL

此属性将复制到所有类型的后续消息中。

缺省选项: 可以指定以下选项以提供缺省副本选项集:

MQCOPY_DEFAULT

此属性将复制到正在转发的消息中，复制到报告消息中，或者复制到订户在发布消息时接收到的消息中。

这相当于指定选项 MQCOPY_FOR 何承天，加 MQCOPY_REPORT 以及 MQCOPY_PUBLISH 的组合。

如果不需要上述任何选项，请使用以下选项:

MQCOPY_NONE

使用此值来指示未指定任何其他复制选项; 以编程方式，此属性与后续消息之间不存在任何关系。对于消息描述符属性，将始终返回此值。

这是 MQSETMP 调用的输入/输出字段以及 MQINQMP 调用的输出字段。此字段的初始值为 MQCOPY_DEFAULT。

选项 (MQLONG)

该值必须为:

MQPD_NONE

未指定选项

这始终是一个输入字段。此字段的初始值为 MQPD_NONE。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQPD_STRUC_ID

属性描述符结构的标识。

对于 C 编程语言，还定义了常量 MQPD_STRUC_ID_ARRAY; 此值与 MQPD_STRUC_ID 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQPD_STRUC_ID。

支持 (MQLONG)

此字段描述队列管理器需要消息属性的支持级别，以便将包含此属性的消息放入队列。这仅适用于 WebSphere MQ 定义的属性; 对所有其他属性的支持是可选的。

当队列管理器知道 WebSphere MQ 定义的属性时，该字段将自动设置为正确的值。如果无法识别该属性，那么将指定 MQPD_SUPPORT_OPTIONAL。当队列管理器接收到包含由队列管理器识别为不正确的 WebSphere MQ 定义的属性的消息时，队列管理器将更正 *Support* 字段的值。

在设置了 MQCMHO_NO_VALIDATION 选项的消息句柄上使用 MQSETMP 调用来设置 WebSphere MQ 定义的属性时，*Support* 将成为输入字段。这允许应用程序使用正确的值放置 WebSphere MQ 定义的属性，其中所连接的队列管理器不支持该属性，但打算在另一个队列管理器上处理该消息。

值 MQPD_SUPPORT_OPTIONAL 始终分配给非 WebSphere MQ 定义的属性。

如果支持消息属性的 WebSphere MQ 版本 7.0 队列管理器接收到包含无法识别的 *Support* 值的属性，那么会将该属性视为如下所示：

- 如果 MQPD_REJECT_UNSUP_MASK 中包含任何无法识别的值，那么指定了 MQPD_SUPPORT_REQUIRED。
- 如果 MQPD_ACCEPT_UNSUP_IF_XMIT_MASK 中包含任何无法识别的值，那么指定了 MQPD_SUPPORT_REQUIRED_IF_LOCAL
- 否则，指定了 MQPD_SUPPORT_OPTIONAL。

在设置了 MQCMHO_NO_VALIDATION 选项的消息句柄上使用 MQSETMP 调用时，MQINQMP 调用将返回下列其中一个值，或者可以指定其中一个值：

MQPD_SUPPORT_OPTIONAL

队列管理器接受该属性，即使该属性不受支持也是如此。可以废弃该属性以使消息流向不支持消息属性的队列管理器。此值还会分配给未由 WebSphere MQ 定义的属性。

MQPD_SUPPORT_REQUIRED

需要对该属性的支持。消息被不支持 WebSphere MQ 定义的属性的队列管理器拒绝。MQPUT 或 MQPUT1 调用失败，完成代码为 MQCC_FAILED，原因码为 MQRC_UNSUPPORTED_PROPERTY。

MQPD_SUPPORT_REQUIRED_IF_LOCAL

如果消息以本地队列为目标，那么该消息将被不支持 WebSphere MQ 定义的属性的队列管理器拒绝。MQPUT 或 MQPUT1 调用失败，完成代码为 MQCC_FAILED，原因码为 MQRC_UNSUPPORTED_PROPERTY。

如果消息以远程队列管理器为目标，那么 MQPUT 或 MQPUT1 调用将成功。

这是 MQINQMP 调用上的输出字段，如果消息句柄是在设置了 MQCMHO_NO_VALIDATION 选项的情况下创建的，那么这是 MQSETMP 调用上的输入字段。此字段的初始值为 MQPD_SUPPORT_OPTIONAL。

Version (MQLONG)

这是结构版本号；值必须为：

MQPD_VERSION_1

Version-1 属性描述符结构。

以下常量指定当前版本的版本号：

MQPD_CURRENT_VERSION

属性描述符结构的当前版本。

这始终是一个输入字段。此字段的初始值为 **MQPD_VERSION_1**。

MQPD 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|--------------------|-----------------------|------|
| <i>StrucId</i> | MQPD_STRUC_ID | 'PD' |
| <i>Version</i> | MQPD_VERSION_1 | 1 |
| <i>Options</i> | MQPD_NONE | 0 |
| <i>Support</i> | MQPD_SUPPORT_OPTIONAL | 0 |
| <i>Context</i> | MQPD_NO_CONTEXT | 0 |
| <i>CopyOptions</i> | MQCOPY_DEFAULT | 0 |

表 524: MQPD 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|--|-------|-----|
| 注意: | | |
| 1. 在 C 编程语言中, 宏变量 MQPD_DEFAULT 包含上面列出的值。它可以通过以下方式用于为结构中的字段提供初始值: | | |
| <pre>MQPD MyPD = {MQPD_DEFAULT};</pre> | | |

C 声明

```
typedef struct tagMQPD MQPD;
struct tagMQPD {
    MQCHAR4   StructId;      /* 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 声明

```
** 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.
```

PL/I 声明

```
dcl
1 MQPD based,
3 StructId 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 */
```

High Level Assembler 声明

```
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-Put-message 选项

下表汇总了结构中的字段。

| 表 525: MQPMO 结构 | | |
|---|--|-----------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 用于控制 MQPUT 和 MQPUT1 的操作的选项 | 选项 |
| <i>Timeout</i> | 已保留 | TIMEOUT |
| <i>Context</i> | 输入队列的对象句柄 | 上下文 |
| <i>KnownDestCount</i> | 成功发送到本地队列的消息数 | KnownDestCount |
| <i>UnknownDestCount</i> | 成功发送到远程队列的消息数 | UnknownDestCount |
| <i>InvalidDestCount</i> | 无法发送的消息数 | InvalidDestCount |
| <i>ResolvedQName</i> | 目标队列的已解析名称 | ResolvedQName |
| <i>ResolvedQMGrName</i> | 目标队列管理器的已解析名称 | ResolvedQMGrName |
| 注: 如果 <i>Version</i> 小于 MQPMO_VERSION_2, 那么将忽略其余字段。 | | |
| <i>RecsPresent</i> | 存在的放置消息记录数或响应记录数 | RecsPresent |
| <i>PutMsgRecFields</i> | 指示哪些 MQPMR 字段存在的标志 | PutMsgRecFields |
| <i>PutMsgRecOffset</i> | 第一个放置消息记录与 MQPMO 的开始处之间的偏移量 | PutMsgRecOffset |
| <i>ResponseRecOffset</i> | 第一个响应记录与 MQPMO 的开始处之间的偏移量 | ResponseRecOffset |
| <i>PutMsgRecPtr</i> | 第一个放置消息记录的地址 | PutMsgRecPtr |
| <i>ResponseRecPtr</i> | 第一个响应记录的地址 | ResponseRecPtr |
| 注: 如果 <i>Version</i> 小于 MQPMO_VERSION_3, 那么将忽略其余字段。 | | |
| <i>OriginalMsgHandle</i> | 原始消息句柄 | OriginalMsgHandle |
| <i>NewMsgHandle</i> | 新建消息句柄 | NewMsgHandle |
| <i>Action</i> | 正在执行的放置的类型以及 <i>OriginalMsgHandle</i> 字段指定的原始消息与 <i>NewMsgHandle</i> 字段指定的新消息之间的关系 | 操作 |
| <i>PubLevel</i> | 出版物所针对的预订级别 | PubLevel |

MQPMO 概述

可用性: 所有 WebSphere MQ 系统, 以及连接到这些系统的 WebSphere MQ 客户机。

用途: MQPMO 结构允许应用程序指定用于控制如何将消息放入队列或发布到主题的选项。此结构是 MQPUT 和 MQPUT1 调用上的输入/输出参数。

版本: MQPMO 的当前版本为 MQPMO_VERSION_3。某些字段仅在特定版本的 MQPMO 中可用。如果需要在多个环境之间移植应用程序, 那么必须确保 MQPMO 的版本在所有环境之间一致。仅在结构的特定版本中存在的字段在 [第 433 页的『MQPMO-Put-message 选项』](#) 和字段描述中标识为此类字段。

为受支持的编程语言提供的头 COPY 和 INCLUDE 文件包含环境支持的 MQPMO 最新版本，但 *Version* 字段的初始值设置为 MQPMO_VERSION_1。要使用 version-1 结构中不存在的字段，应用程序必须将 *Version* 字段设置为所需版本的版本号。

字符集和编码:MQPMO 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是，如果应用程序作为 MQ MQI 客户机运行，那么该结构必须采用客户机的字符集和编码。

MQPMO 的字段

MQPMO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

操作 (MQLONG)

这指定要执行的放置的类型以及由 *OriginalMsgHandle* 字段指定的原始消息与由 *NewMsgHandle* 字段指定的新消息之间的关系。消息的属性由队列管理器根据指定的操作的值选择。

您可以选择在 MQPUT 或 MQPUT1 调用上使用 *MsgDesc* 参数来提供消息描述符的内容。或者，可以不提供 *MsgDesc* 参数，也可以通过在 MQPMO 结构的 "选项" 字段中包含 MQPMO_MD_FOR_OUTPUT_ONLY 来指定仅输出该参数。

如果未提供 *MsgDesc* 参数，或者如果将其指定为仅输出，那么将根据本主题中描述的规则从 MQPMO 的消息句柄字段填充新消息的消息描述符。

在组成消息描述符后，[控制上下文信息](#) 中描述的上下文设置和传递活动将生效。

如果指定了不正确的操作值，那么调用将失败，原因码为 MQRC_ACTION_ERROR。

可以指定下列任何一项操作:

MQACTP_NEW

正在放置新消息，并且程序未指定与先前消息的关系。消息描述符组成如下:

- 如果在 MQPUT 或 MQPUT1 调用上提供了 *MsgDesc*，并且 MQPMO_MD_FOR_OUTPUT_ONLY 不在 MQPMO.Options，此选项用作未修改的消息描述符。
- 如果未提供 *MsgDesc*，或者 MQPMO_MD_FOR_OUTPUT_ONLY 位于 MQPMO.Options 然后，队列管理器使用来自 *OriginalMsgHandle* 和 *NewMsgHandle* 的属性组合来生成消息描述符。在新消息句柄上显式设置的任何消息描述符字段都优先于原始消息句柄中的那些字段。

消息数据取自 MQPUT 或 MQPUT1 缓冲区参数。

MQACTP_FORWARD

正在转发先前检索的消息。原始消息句柄指定先前检索的消息。

新消息句柄指定对原始消息句柄中的属性 (包括消息描述符中的任何属性) 的任何修改。

消息描述符组成如下:

- 如果在 MQPUT 或 MQPUT1 调用上提供了 *MsgDesc*，并且 MQPMO_MD_FOR_OUTPUT_ONLY 不在 MQPMO.Options，此选项用作未修改的消息描述符。
- 如果未提供 *MsgDesc*，或者 MQPMO_MD_FOR_OUTPUT_ONLY 位于 MQPMO.Options 然后，队列管理器使用来自 *OriginalMsgHandle* 和 *NewMsgHandle* 的属性组合来生成消息描述符。在新消息句柄上显式设置的任何消息描述符字段都优先于原始消息句柄中的那些字段。
- 如果在 MQPMO.Options，那么将遵守这些选项。

消息属性组成如下:

- 原始消息句柄中在 MQPD.CopyOptions
- 来自新消息句柄的所有属性。对于新消息句柄中与原始消息句柄中的属性同名的每个属性，将从新消息句柄中获取值。此规则的唯一例外情况是，当新消息句柄中的属性与原始消息句柄中的属性同名，但该属性的值为空时。在这种情况下，将从消息中除去该属性。

要转发的消息数据取自 MQPUT 或 MQPUT1 缓冲区参数。

MQACTP_REPLY

正在对先前检索的消息进行应答。原始消息句柄指定先前检索的消息。

新消息句柄指定对原始消息句柄中的属性 (包括消息描述符中的任何属性) 的任何修改。

消息描述符组成如下:

- 如果在 MQPUT 或 MQPUT1 调用上提供了 MsgDesc , 并且 MQPMO_MD_FOR_OUTPUT_ONLY 不在 MQPMO.Options, 此选项用作未修改的消息描述符。
- 如果未提供 MsgDesc , 或者 MQPMO_MD_FOR_OUTPUT_ONLY 位于 MQPMO.Options 然后按如下所示选择初始消息描述符字段:

| 表 526: 应答消息句柄变换 | |
|-----------------|--|
| MQMD 中的字段 | 使用的值 |
| 报告 | 如果 MQRO_PASS_DISCARD_AND_到期 设置了 MQRO_DISCARD_MSG: MQRO_DISCARD_MSG 否则 MQRO_NONE |
| MsgType | MQMT_REPLY |
| 到期 | 如果 MQRO_PASS_DISCARD_AND_到期 已设置: 从输入消息复制 否则 MQEI_UNLIMITED |
| Feedback | MQFB_NONE |
| MsgId | 如果设置了 MQPMO_NEW_MSG_ID: 生成新的消息标识 否则, 如果设置了 MQRO_PASS_MSG_ID: 从输入消息复制 否则 MQMI_NONE |
| CorrelId | 如果设置了 MQPMO_NEW_CORREL_ID: 生成新的相关标识 否则, 如果设置了 MQRO_COPY_MSG_ID_TO_CORREL_ID: 从 MsgId 字段复制 输入消息 else 如果设置了 MQRO_PASS_CORREL_ID: 从 CorrelId 字段复制 输入消息 否则 MQCI_NONE |
| BackoutCount | 0 |
| ReplyToQ | 空白 |
| ReplyToQMgr | 空白 |
| GroupId | MQGI_NONE |
| MsgSeqNumber | 1 |
| 偏移量 | 0 |
| MsgFlags | MQMF_NONE |

| 表 526: 应答消息句柄变换 (继续) | |
|----------------------|----------------|
| MQMD 中的字段 | 使用的值 |
| OriginalLength | MQOL_UNDEFINED |

- 然后，新消息句柄将修改消息描述符-任何显式设置为新消息句柄中的属性的消息描述符字段优先于上述消息描述符字段。

消息属性组成如下:

- 原始消息句柄中在 MQPD.CopyOptions
- 来自新消息句柄的所有属性。对于新消息句柄中与原始消息句柄中的属性同名的每个属性，将从新消息句柄中获取值。此规则的唯一例外情况是，当新消息句柄中的属性与原始消息句柄中的属性同名，但该属性的值为空时。在这种情况下，将从消息中除去该属性。

要转发的消息数据取自 MQPUT/MQPUT1 缓冲区参数。

MQACTP_REPORT

由于先前检索的消息，正在生成报告。原始消息句柄指定导致生成报告的消息。

新消息句柄指定对原始消息句柄中的属性 (包括消息描述符中的任何属性) 的任何修改。

消息描述符组成如下:

- 如果在 MQPUT 或 MQPUT1 调用上提供了 MsgDesc，并且 MQPMO_MD_FOR_OUTPUT_ONLY 不在 MQPMO.Options，此选项用作未修改的消息描述符。
- 如果未提供 MsgDesc，或者 MQPMO_MD_FOR_OUTPUT_ONLY 位于 MQPMO.Options 然后按如下所示选择初始消息描述符字段:

| 表 527: 报告消息句柄变换 | |
|-----------------|---|
| MQMD 中的字段 | 使用的值 |
| 报告 | 如果 MQRO_PASS_DISCARD_AND_到期和已设置 MQRO_DISCARD_MSG: MQRO_DISCARD_MSG 否则 MQRO_NONE |
| MsgType | MQMT_REPORT |
| 到期 | 如果 MQRO_PASS_DISCARD_AND_到期已设置: 从输入消息复制 否则 MQEI_UNLIMITED |
| MsgId | 如果设置了 MQPMO_NEW_MSG_ID: 生成新的消息标识 否则，如果设置了 MQRO_PASS_MSG_ID: 从输入消息复制 否则 MQMI_NONE |

表 527: 报告消息句柄变换 (继续)

| MQMD 中的字段 | 使用的值 |
|----------------|--|
| CorrelId | 如果设置了 MQPMO_NEW_CORREL_ID: 生成新的相关标识 否则, 如果设置了 MQRO_COPY_MSG_ID_TO_CORREL_ID: 从 MsgId 字段复制 输入消息 else 如果设置了 MQRO_PASS_CORREL_ID: 从 CorrelId 字段复制 输入消息 否则 MQCI_NONE |
| BackoutCount | 0 |
| ReplyToQ | 空白 |
| ReplyToQMgr | 空白 |
| OriginalLength | 设置为 <i>BufferLength</i> |

- 然后, 新消息句柄将修改消息描述符-任何显式设置为新消息句柄中的属性的消息描述符字段优先于上述消息描述符字段。

消息属性组成如下:

- 原始消息句柄中在 MQPD.CopyOptions
- 来自新消息句柄的所有属性。对于新消息句柄中与原始消息句柄中的属性同名的每个属性, 将从新消息句柄中获取值。此规则的唯一例外情况是, 当新消息句柄中的属性与原始消息句柄中的属性同名, 但该属性的值为空时。在这种情况下, 将从消息中除去该属性。

生成的 MQMD 中的 "反馈" 字段表示要生成的报告。反馈值 MQFB_NONE 导致 MQPUT 或 MQPUT1 调用失败, 原因码为 MQRC_FEEDBACK_ERROR。

要选择报告消息的用户数据, WebSphere MQ 将查阅生成的 MQMD 中的 "报告" 和 "反馈" 字段, 以及 MQPUT 或 MQPUT1 调用的 "缓冲区" 和 BufferLength 参数。

- 如果 "反馈" 为 MQFB_COA, MQFB_COD 或 MQFB_EXPIRATION, 那么将检查 "报告" 的值。
- 如果下列任何情况成立, 那么将使用来自 Buffer 的长度为 BufferLength 的完整消息数据。
 - 反馈为 MQFB_EXPIRATION, 报告包含 MQRO_EXPIRATION_WITH_FULL_DATA
 - 反馈为 MQFB_COD, 报告包含 MQRO_COD_WITH_FULL_DATA
 - 反馈为 MQFB_COA, 报告包含 MQRO_COA_WITH_FULL_DATA
- 如果以下任何情况成立, 那么将使用来自 Buffer 的消息的前 100 个字节 (如果此值小于 100, 那么为 BufferLength)
 - 反馈为 MQFB_EXPIRATION, 报告包含 MQRO_EXPIRATION_WITH_DATA
 - 反馈为 MQFB_COD, 报告包含 MQRO_COD_WITH_DATA
 - 反馈为 MQFB_COA, 报告包含 MQRO_COA_WITH_DATA
- 如果 "反馈" 为 MQFB_EXPIRATION, MQFB_COD 或 MQFB_COA, 并且 "报告" 不包含与该 Feedback 值相关的 *_WITH_FULL_DATA 或 *_WITH_DATA 选项, 那么消息中不会包含任何用户数据。
- 如果 "反馈" 采用与上面列出的值不同的值, 那么将正常使用 Buffer 和 BufferLength。

下表中显示了用户数据的派生:

| 表 528: 用户数据的来源 | | | |
|---------------------------------------|-----------------|-----------------|------------------------|
| | MQFB_COA | MQFB_COD | MQFB_EXPIRATION |
| MQRO_EXPIRATION_WITH_FULL_DATA | None | None | 缓冲区 (缓冲区长度) |
| MQRO_COD_WITH_FULL_DATA | None | 缓冲区 (缓冲区长度) | None |
| MQRO_COA_WITH_FULL_DATA | 缓冲区 (缓冲区长度) | None | None |
| MQRO_EXPIRATION_WITH_DATA | None | None | 缓冲区 (前 100 个字节) |
| MQRO_COD_WITH_DATA | None | 缓冲区 (前 100 个字节) | None |
| MQRO_COA_WITH_DATA | 缓冲区 (前 100 个字节) | None | None |

上下文 (MQHOBJS)

如果指定了 MQPMO_PASS_IDENTITY_CONTEXT 或 MQPMO_PASS_ALL_CONTEXT，那么此字段必须包含从中获取要与所放入的消息相关联的上下文信息的输入队列句柄。

如果既未指定 MQPMO_PASS_IDENTITY_CONTEXT，也未指定 MQPMO_PASS_ALL_CONTEXT，那么将忽略此字段。

这是一个输入字段。此字段的初始值为 0。

InvalidDest 计数 (MQLONG)

这是无法发送到分发列表中的队列的消息数。计数包括未能打开的队列，以及已成功打开但放置操作失败的队列。将消息放入不在分发列表中的单个队列时，也会设置此字段。

注: 如果 MQPUT 或 MQPUT1 调用上的 *CompCode* 参数为 MQCC_OK 或 MQCC_WARNN; 如果 *CompCode* 参数为 MQCC_FAILED，但在应用程序代码中不依赖于此参数，那么可以设置此字段。

这是输出字段。此字段的初始值为 0。如果 *Version* 小于 MQPMO_VERSION_1，那么不会设置此字段。

在 z/OS 上未定义此字段，因为不支持分发列表。

KnownDest 计数 (MQLONG)

这是当前 MQPUT 或 MQPUT1 调用成功发送到作为本地队列的分发列表中的队列的消息数。该计数不包括发送到解析为远程队列的队列的消息 (即使最初使用本地传输队列来存储消息)。将消息放入不在分发列表中的单个队列时，也会设置此字段。

这是输出字段。此字段的初始值为 0。如果 *Version* 小于 MQPMO_VERSION_1，那么不会设置此字段。

在 z/OS 上未定义此字段，因为不支持分发列表。

NewMsg 句柄 (MQHMSG)

这是受 "操作" 字段值限制的消息的可选句柄。它定义消息的属性并覆盖 *OriginalMsgHandle* 的值 (如果已指定)。

从 MQPUT 或 MQPUT1 调用返回时，句柄的内容将反映实际放入的消息。

这是一个输入字段。此字段的初始值为 MQHM_NONE。如果版本小于 MQPMO_VERSION_3，那么将忽略此字段。

MQPMO 选项 (MQLONG)

"选项" 字段控制 MQPUT 和 MQPUT1 调用的操作。

作用域选项。 可以指定任何 MQPMO 选项，也可以不指定任何 MQPMO 选项。如果需要多个选项，那么可以通过以下方式使用您为选项指定的值：

- 可以添加值。请勿多次添加同一常量。
- 如果编程语言支持按位运算，那么可以使用按位 OR 运算来组合值。

将记录无效的组合；任何其他组合都有效。

以下选项控制发送的发布的范围：

MQPMO_SCOPE_QMGR

发布仅发送给已在此队列管理器上预订的订户。发布不会转发到任何已预订此队列管理器的远程发布/预订队列管理器，这将覆盖使用 PUBSCOPE 主题属性设置的任何行为。

注：如果未设置，那么发布范围由 PUBSCOPE 主题属性确定。

发布选项。 以下选项控制将消息发布到主题的方式：

MQPMO_SUPPRESS_REPLYTO

此出版物的 MQMD 的 *ReplyToQ* 和 *ReplyToQMGR* 字段中指定的任何信息都不会传递给订户。如果此选项与需要 *ReplyToQ* 的报告选项一起使用，那么调用将失败并返回 MQRC_MISSING_REPLY_TO_Q。

MQPMO_RETAIN

要发送的发布将由队列管理器保留。此保留时间允许订户使用 MQSUBRQ 调用在发布此发布后请求此发布的副本。它还允许将发布发送到在创建此发布之后进行其预订的应用程序（除非他们选择不使用选项 MQSO_NEW_publicATIONS_ONLY 发送此发布）。如果向应用程序发送了保留的发布，那么该发布由该发布的 MQIsRetained 消息属性指示。

在主题树的每个节点上只能保留一个发布内容。因此，如果已存在此主题的保留出版物（由任何其他应用程序发布），那么会将其替换为此出版物。因此，最好避免有多个发布程序保留同一主题上的消息。

当订户请求保留发布时，所使用的预订可能在主题中包含通配符，在这种情况下，许多保留发布可能匹配（在主题树中的各个节点上），并且可能会将多个发布发送到请求应用程序。请参阅 [第 698 页的『MQSUBRQ-预订请求』](#) 调用的描述以获取更多详细信息。

有关保留发布如何与预订级别交互的信息，请参阅 [拦截发布](#)。

如果使用此选项并且无法保留发布，那么将不会发布消息，并且调用将失败并返回 MQRC_PUT_NOT_保留。

MQPMO_NOT_OWN_SUBS

告知队列管理器，应用程序不希望将其任何发布发送到它拥有的预订。如果连接句柄相同，那么会将预订视为由同一应用程序拥有。

MQPMO_WARN_IF_NO_SUBS_MATCHED

如果没有预订与发布内容匹配，请返回完成代码 (*CompCode*) MQCC_WARNING 和原因码 MQRC_NO_SUBS_MATCHES。

如果 put 操作返回了 MQRC_NO_SUBS_MODIFIED，那么发布未传递到任何预订。但是，如果在 put 操作指定了 MQPMO_RETAIN 选项，那么将保留该消息并将其传递到随后定义的任何匹配预订。

如果满足以下任何条件，那么主题上的预订将与发布匹配：

- 消息已传递到预订队列
- 消息将已传递到预订队列，但队列存在问题意味着无法将消息放入队列，因此将其放置在死信队列上或将其废弃。
- 定义了路由出口，禁止将消息传递到预订

如果满足以下任何条件，那么该主题的预订与发布不匹配：

- 预订具有与发布不匹配的选择字符串
- 预订指定了 MQSO_PUBLICICATION_ON_REQUEST 选项
- 未交付发布内容，因为在 put 操作指定了 MQPMO_NOT_OWN_SUBS 选项，并且预订与发布者的身份相匹配

同步点选项。 以下选项与 MQPUT 或 MQPUT1 调用在工作单元中的参与相关:

MQPMO_SYNCPOINT

请求是在正常工作单元协议中运行。在落实工作单元之前, 该消息在工作单元外部不可见。如果工作单元已回退, 那么将删除该消息。

如果未指定 MQPMO_SYNCPOINT 和 MQPMO_NO_SYNCPOINT, 那么在工作单元协议中包含 put 请求由运行队列管理器的环境而不是运行应用程序的环境确定。在 z/OS 上, put 请求位于工作单元中。在所有其他环境中, put 请求不在工作单元中。

由于这些差异, 要移植的应用程序不得允许此选项为缺省值; 请显式指定 MQPMO_SYNCPOINT 或 MQPMO_NO_SYNCPOINT。

请勿将 MQPMO_SYNCPOINT 与 MQPMO_NO_SYNCPOINT 一起指定。

MQPMO_NO_SYNCPOINT

请求是在正常工作单元协议之外运行。消息立即可用, 无法通过回退工作单元将其删除。

如果未指定 MQPMO_NO_SYNCPOINT 和 MQPMO_SYNCPOINT, 那么在工作单元协议中包含 put 请求由运行队列管理器的环境而不是运行应用程序的环境确定。在 z/OS 上, put 请求位于工作单元中。在所有其他环境中, put 请求不在工作单元中。

由于这些差异, 要移植的应用程序不得允许此选项为缺省值; 请显式指定 MQPMO_SYNCPOINT 或 MQPMO_NO_SYNCPOINT。

请勿将 MQPMO_NO_SYNCPOINT 与 MQPMO_SYNCPOINT 一起指定。

消息标识和相关标识选项。 以下选项请求队列管理器生成新的消息标识或相关标识:

MQPMO_NEW_MSG_ID

队列管理器将 MQMD 中 *MsgId* 字段的内容替换为新的消息标识。此消息标识随消息一起发送, 并在 MQPUT 或 MQPUT1 调用的输出时返回到应用程序。

将消息放入分发列表时, 也可以指定 MQPMO_NEW_MSG_ID 选项; 请参阅 MQPMR 结构中 *MsgId* 字段的描述以获取详细信息。

使用此选项将使应用程序无需在每次 MQPUT 或 MQPUT1 调用之前将 *MsgId* 字段重置为 MQMI_NONE。

MQPMO_NEW_CORREL_ID

队列管理器将 MQMD 中 *CorrelId* 字段的内容替换为新的相关标识。此相关标识随消息一起发送, 并在 MQPUT 或 MQPUT1 调用的输出时返回到应用程序。

在将消息放入分发列表时, 还可以指定 MQPMO_NEW_CORREL_ID 选项; 请参阅 MQPMR 结构中 *CorrelId* 字段的描述以获取详细信息。

MQPMO_NEW_CORREL_ID 在应用程序需要唯一相关标识的情况下很有用。

组和段选项。 以下选项与处理逻辑消息的组和段中的消息相关。请阅读以下定义, 以帮助了解该选项。



注意: 不能为发布/预订使用分段或组合消息。

物理消息

这是可以放在队列上或从队列中除去的最小信息单元; 它通常对应于在单个 MQPUT, MQPUT1 或 MQGET 调用上指定或检索的信息。每条物理消息都有自己的消息描述符 (MQMD)。通常, 物理消息由消息标识 (MQMD 中的 *MsgId* 字段) 的不同值进行区分, 尽管队列管理器未实施此操作。

逻辑消息

逻辑消息是仅适用于非 z/OS 平台的单个应用程序信息单元。在没有系统约束的情况下, 逻辑消息与物理消息相同。但是, 如果逻辑消息非常大, 那么系统约束可能建议或需要将逻辑消息拆分为两个或多个物理消息 (称为段)。

已分段的逻辑消息由具有相同非空组标识 (MQMD 中的 *GroupId* 字段) 和相同消息序号 (MQMD 中的 *MsgSeqNumber* 字段) 的两个或多个物理消息组成。这些段通过段偏移量 (MQMD 中的 *Offset* 字段) 的不同值进行区分, 这将提供物理消息中的数据从逻辑消息中的数据开始的偏移量。由于每个段都是物理消息, 因此逻辑消息中的段通常具有不同的消息标识。

未分段但发送应用程序已允许分段的逻辑消息也具有非空组标识，尽管在这种情况下，如果逻辑消息不属于消息组，那么只有一条具有该组标识的物理消息。发送应用程序已禁止分段的逻辑消息具有空组标识 (MQGI_NONE)，除非逻辑消息属于消息组。

消息组

消息组是一组具有相同非空组标识的一个或多个逻辑消息。该组中的逻辑消息由消息序号的不同值区分，该值是 1 到 n 范围内的整数，其中 n 是该组中的逻辑消息数。如果对一个或多个逻辑消息进行分段，那么组中有多个 n 物理消息。

MQPMO_LOGICAL_ORDER

此选项告诉队列管理器应用程序如何将消息放入逻辑消息的组和段中。只能对 MQPUT 调用指定此选项；它对于 MQPUT1 调用是无效的。

如果指定 MQPMO_LOGICAL_ORDER，那么指示应用程序使用连续的 MQPUT 调用来执行以下操作：

1. 按段偏移量的递增顺序（从 0 开始，无间隔）放置每条逻辑消息中的段。
2. 先放置一条逻辑消息中的所有段，然后再放置下一条逻辑消息中的段。
3. 按消息序号的递增顺序（从 1 开始，无间隔）放置每个消息组中的逻辑消息。IBM WebSphere MQ 自动递增消息序号。
4. 先放置一个消息组中的所有逻辑消息，然后再放置下一个消息组中的逻辑消息。

有关 MQPMO_LOGICAL_ORDER 的详细信息，请参阅 [逻辑和物理排序](#)

上下文选项。 以下选项控制消息上下文的处理：

MQPMO_NO_CONTEXT

标识和源上下文都设置为指示无上下文。这意味着 MQMD 中的上下文字段设置为：

- 字符字段的空白
- 字节字段的空值
- 数字字段的零

MQPMO_DEFAULT_CONTEXT

对于身份和源，此消息将具有与其关联的缺省上下文信息。队列管理器按如下所示设置消息描述符中的上下文字段：

| MQMD 中的字段 | 使用的值 |
|-------------------------|---------------------------------|
| <i>UserIdentifier</i> | 如果可能，根据环境确定；否则设置为空白。 |
| <i>AccountingToken</i> | 根据环境确定 (如果可能)；否则设置为 MQACT_NONE。 |
| <i>ApplIdentityData</i> | 设置为空白。 |
| <i>PutApplType</i> | 从环境中确定。 |
| <i>PutApplName</i> | 如果可能，根据环境确定；否则设置为空白。 |
| <i>PutDate</i> | 设置为放入消息的日期。 |
| <i>PutTime</i> | 设置为放入消息的时间。 |
| <i>ApplOriginData</i> | 设置为空白。 |

有关消息上下文的更多信息，请参阅 [消息上下文](#)。

如果未指定任何上下文选项，那么这些是缺省值和操作。

MQPMO_PASS_IDENTITY_CONTEXT

该消息将具有与其关联的上下文信息。身份上下文取自 *Context* 字段中指定的队列句柄。源上下文信息由队列管理器以与 MQPMO_DEFAULT_CONTEXT 相同的方式生成 (请参阅上表以了解值)。有关消息上下文的更多信息，请参阅 [消息上下文](#)。

对于 MQPUT 调用，必须已使用 MQOO_PASS_IDENTITY_CONTEXT 选项 (或暗示该选项的选项) 打开队列。对于 MQPUT1 调用，将执行与使用 MQOO_PASS_IDENTITY_CONTEXT 选项的 MQOPEN 调用相同的授权检查。

MQPMO_PASS_ALL_CONTEXT

该消息将具有与其关联的上下文信息。上下文取自 *Context* 字段中指定的队列句柄。有关消息上下文的更多信息，请参阅 [控制上下文信息](#)。

对于 MQPUT 调用，必须已使用 MQOO_PASS_ALL_CONTEXT 选项 (或暗示该选项的选项) 打开队列。对于 MQPUT1 调用，将执行与使用 MQOO_PASS_ALL_CONTEXT 选项的 MQOPEN 调用相同的授权检查。

MQPMO_SET_IDENTITY_CONTEXT

该消息将具有与其关联的上下文信息。应用程序在 MQMD 结构中指定身份上下文。源上下文信息由队列管理器以与 MQPMO_DEFAULT_CONTEXT 相同的方式生成 (请参阅上表以了解值)。有关消息上下文的更多信息，请参阅 [消息上下文](#)。

对于 MQPUT 调用，必须已使用 MQOO_SET_IDENTITY_CONTEXT 选项 (或暗示该选项的选项) 打开队列。对于 MQPUT1 调用，将执行与使用 MQOO_SET_IDENTITY_CONTEXT 选项的 MQOPEN 调用相同的授权检查。

MQPMO_SET_ALL_CONTEXT

该消息将具有与其关联的上下文信息。应用程序在 MQMD 结构中指定身份、源和用户上下文。有关消息上下文的更多信息，请参阅 [消息上下文](#)。

对于 MQPUT 调用，必须已使用 MQOO_SET_ALL_CONTEXT 选项打开队列。对于 MQPUT1 调用，将执行与使用 MQOO_SET_ALL_CONTEXT 选项的 MQOPEN 调用相同的授权检查。

只能指定其中一个 MQPMO_*_CONTEXT 上下文选项。如果未指定任何值，那么将采用 MQPMO_DEFAULT_CONTEXT。

属性选项。 以下选项与消息的属性相关：

MQPMO_MD_FOR_OUTPUT_ONLY

必须仅将消息描述符参数用于输出以返回放入的消息的消息描述符。必须将与 **MQPMO** 结构的 *NewMsgHandle* 和/或 *OriginalMsgHandle* 字段相关联的消息描述符字段用于输入。

如果未提供有效的消息句柄，那么调用将失败，原因码为 **MQRC_MD_ERROR**。

放置响应选项。 以下选项控制返回到 MQPUT 或 MQPUT1 调用的响应。只能指定其中一个选项。如果未指定 MQPMO_ASYNC_RESPONSE 和 MQPMO_SYNC_RESPONSE，那么将采用 MQPMO_RESPONSE_AS_Q_DEF 或 MQPMO_RESPONSE_AS_TOPIC_DEF。

MQPMO_ASYNC_RESPONSE

MQPMO_ASYNC_RESPONSE 选项请求完成 MQPUT 或 MQPUT1 操作，而无需应用程序等待队列管理器完成调用。使用此选项可以提高消息传递性能，尤其是对于使用客户机绑定的应用程序。应用程序可以使用 MQSTAT 动词定期检查在任何先前异步调用期间是否发生了错误。

使用此选项时，仅保证在 MQMD 中完成以下字段：

- ApplIdentityData
- PutApplType
- PutApplName
- ApplOriginData

此外，如果将 MQPMO_NEW_MSG_ID 或 MQPMO_NEW_CORREL_ID 指定为选项，那么还将完成返回的 *MsgId* 和 *CorrelId*。(可以通过指定空白 *MsgId* 字段来隐式指定 MQPMO_NEW_MSG_ID)。

仅完成前面指定的字段。未定义通常在 MQMD 或 MQPMO 结构中返回的其他信息。

请求 MQPUT1 的异步放置响应时，未定义 MQOD 结构中返回的 *ResolvedQName* 和 *ResolvedQMgr* 名称。

当针对 MQPUT 或 MQPUT1 请求异步放入响应时，*CompCode* 和 *Reason* 为 MQCC_OK 和 MQRC_NONE 不一定意味着消息已被成功放入队列中。开发使用异步 put 响应的 MQI 应用程序时，如果需要确认消息已放入队列，那么必须从 put 操作中同时检查 *CompCode* 和 *Reason* 代码，并使用 MQSTAT 来查询异步错误信息。

虽然不会立即返回每个单独 MQPUT 或 MQPUT1 调用的成功或失败，但稍后可以通过调用 MQSTAT 来确定异步调用下发生的第一个错误。

如果使用异步 put 响应未能传递同步点下的持久消息，并且您尝试落实该事务，那么落实将失败，并且将使用完成代码 MQCC_FAILED 和原因 MQRC_BACKED_OUT 回退该事务。应用程序可以调用 MQSTAT 以确定先前 MQPUT 或 MQPUT1 失败的原因。

MQPMO_SYNC_RESPONSE

指定此 put 响应类型可确保始终同步发出 MQPUT 或 MQPUT1 操作。如果 put 操作成功，那么将完成 MQMD 和 MQPMO 中的所有字段。

此选项确保同步响应，而不考虑在队列或主题对象上定义的缺省 put 响应值。

MQPMO_RESPONSE_AS_Q_DEF

如果为 MQPUT 调用指定此值，那么所使用的 put 响应类型将从应用程序首次打开队列时在队列上指定的 DEFpresP 值中获取。如果客户机应用程序连接到版本低于 7.0 的级别的队列管理器，那么它的行为就像指定了 MQPMO_SYNC_RESPONSE 一样。

如果为 MQPUT1 调用指定了此选项，那么在将请求发送到服务器之前，DEFpresP 属性的值未知。缺省情况下，如果 MQPUT1 调用使用的是 MQPMO_SYNCPOINT，那么它的行为与 MQPMO_ASYNC_RESPONSE 一样，如果使用的是 MQPMO_NO_SYNCPOINT，那么它的行为与 MQPMO_SYNC_RESPONSE 相同。但是，您可以通过在客户机配置文件中设置 Put1DefaultAlwaysSync 属性来覆盖此缺省行为，请参阅 [客户机配置文件的 CHANNELS 节](#)。

MQPMO_RESPONSE_AS_TOPIC_DEF

MQPMO_RESPONSE_AS_TOPIC_DEF 是 MQPMO_RESPONSE_AS_Q_DEF 的同义词，用于与主题对象配合使用。

其他选项。 以下选项控制授权检查，队列管理器停顿时发生的情况以及解析队列和队列管理器名称：

MQPMO_ALTERNATE_USER_AUTHORITY

MQPMO_ALTERNATE_USER_AUTHORITY 指示 MQPUT1 调用的 *ObjDesc* 参数中的 *AlternateUserId* 字段包含用于验证将消息放入队列的权限的用户标识。仅当授权 *AlternateUserId* 使用指定的选项打开队列时，调用才能成功，而不管运行应用程序的用户标识是否有权执行此操作。（这不适用于指定的上下文选项，但这些选项始终根据运行应用程序的用户标识进行检查。）

此选项仅对 MQPUT1 调用有效。

MQPMO_FAIL_IF QUIESCING

如果队列管理器处于停顿状态，那么此选项会强制 MQPUT 或 MQPUT1 调用失败。

在 z/OS 上，如果连接（对于 CICS 或 IMS 应用程序）处于停顿状态，那么此选项还会强制 MQPUT 或 MQPUT1 调用失败。

调用返回完成代码 MQCC_FAILED，原因码为 MQRC_Q_MGR QUIESCING 或 MQRC_CONNECTION QUIESCING。

MQPMO_RESOLVE_LOCAL_Q

使用此选项在 MQPMO 结构中填充 *ResolvedQName*，其中包含要将消息放入的本地队列的名称，并在 *ResolvedQMgrName* 中填充托管本地队列的本地队列管理器的名称。有关 MQPMO_RESOLVE_LOCAL_Q 的更多信息，请参阅主题 [MQOO_RESOLVE_LOCAL_Q](#)。

如果您有权放入队列，那么您具有在 MQPUT 调用上指定此标志的必需权限；不需要特殊权限。

缺省选项。 如果不需要所描述的任何选项，请使用以下选项：

MQPMO_NONE

使用此值来指示未指定任何其他选项；所有选项均采用其缺省值。MQPMO_NONE 定义为帮助程序文档；不打算将此选项与任何其他选项一起使用，但由于其值为零，因此无法检测到此类使用。

MQPMO_NONE 是输入字段。Options 字段的初始值为 MQPMO_NONE。

OriginalMsg 句柄 (MQHMSG)

这是消息的可选句柄。它可能是先前从队列中检索到的。此句柄的使用取决于 *Action* 字段的值；另请参阅 [NewMsgHandle](#)。

MQPUT 或 MQPUT1 调用不会改变原始消息句柄的内容。

这是一个输入字段。此字段的初始值为 **MQHM_NONE**。如果版本小于 **MQPMO_VERSION_3**，那么将忽略此字段。

PubLevel (MQLONG)

此字段的初始值为 9。此出版物以预订为目标的级别。只有具有小于或等于此值的最高 **SubLevel** 的预订才会接收此发布内容。此值必须在 0 到 9 的范围内；零是最低级别。但是，如果保留了某个发布，那么它将不再可供更高级别的订户使用，因为它将在 **PubLevel 1** 上重新发布。

有关信息，请参阅 [拦截发布](#)。

PutMsgRecFields (MQLONG)

此字段包含指示应用程序提供的放置消息记录中存在哪些 **MQPMR** 字段的标志。仅当将消息放入分发列表时，才使用 *PutMsgRecFields*。如果 **RecsPresent** 为零，或者 *PutMsgRecOffset* 和 *PutMsgRecPtr* 均为零，那么将忽略该字段。

对于存在的字段，队列管理器将相应放置消息记录中的字段中的值用于每个目标。对于不存在的字段，队列管理器使用 **MQMD** 结构中的值。

使用下列一个或多个标志来指示放置消息记录中存在哪些字段：

MQPMRF_MSG_ID

存在消息标识字段。

MQPMRF_CORREL_ID

存在相关标识字段。

MQPMRF_GROUP_ID

存在组标识字段。

MQPMRF_FEEDBACK

存在反馈字段。

MQPMRF_ACCOUNTING_TOKEN

存在记帐标记字段。

如果指定此标志，请在 *Options* 字段中指定 **MQPMO_SET_IDENTITY_CONTEXT** 或 **MQPMO_SET_ALL_CONTEXT**；如果未满足此条件，那么调用将失败，原因码为 **MQRC_PMO_RECORD_FLAGS_ERROR**。

如果不存在 **MQPMR** 字段，那么可以指定以下内容：

MQPMRF_NONE

不存在放置消息记录字段。

如果指定了此值，那么 **RecsPresent** 必须为零，或者 *PutMsgRecOffset* 和 *PutMsgRecPtr* 都必须为零。

定义 **MQPMRF_NONE** 以帮助程序文档。不打算将此常量与任何其他常量一起使用，但由于其值为零，因此无法检测到此类使用。

如果 *PutMsgRecFields* 包含无效的标志，或者提供了放置消息记录，但 *PutMsgRecFields* 具有值 **MQPMRF_NONE**，那么调用将失败，原因码为 **MQRC_PMO_RECORD_FLAGS_ERROR**。

这是一个输入字段。此字段的初始值为 **MQPMRF_NONE**。如果 *Version* 小于 **MQPMO_VERSION_2**，那么将忽略此字段。

PutMsgRecOffset (MQLONG)

这是从 **MQPMO** 结构开始的第一个 **MQPMR** 放置消息记录的偏移量（以字节为单位）。偏移可以是正数或负数。仅当将消息放入分发列表时，才会使用 *PutMsgRecOffset*。如果 **RecsPresent** 为零，那么将忽略该字段。

将消息放入分发列表时，可以提供一个或多个 **MQPMR** 放置消息记录的数组，以便分别为每个目标指定消息的特定属性；这些属性为：

- 消息标识
- 相关标识

- 组标识
- 反馈值
- 记帐标记

您不需要指定所有这些属性，但无论您选择哪个子集，都以正确的顺序指定字段。请参阅 MQPMR 结构的描述以获取更多详细信息。

通常，打开分发列表时，必须有与 MQOD 指定的对象记录一样多的放置消息记录；每个放置消息记录都提供相应对象记录所标识的队列的消息属性。分发列表中未能打开的队列仍必须将为它们分配的消息记录放在数组中的相应位置，尽管在这种情况下将忽略消息属性。

放入消息记录数可能与对象记录数不同。如果放入消息记录少于对象记录，那么将从消息描述符 MQMD 中的相应字段中获取没有放入消息记录的目标的消息属性。如果存在比对象记录更多的放置消息记录，那么不会使用多余的消息记录（尽管仍必须可以访问这些消息记录）。放置消息记录是可选的，但如果提供了这些记录，那么其中必须有 *RecsPresent* 个记录。

通过在 *PutMsgRecOffset* 中指定偏移量或在 *PutMsgRecPtr* 中指定地址，以类似于 MQOD 中的对象记录的方式提供放置消息记录；有关如何执行此操作的详细信息，请参阅第 414 页的『MQOD-对象描述符』中描述的 *ObjectRecOffset* 字段。

不能使用多个 *PutMsgRecOffset* 和 *PutMsgRecPtr*；调用失败，原因码为 MQRC_PUT_MSG_RECORDS_ERROR（如果两者都非零）。

这是一个输入字段。此字段的初始值为 0。如果 *Version* 小于 MQPMO_VERSION_2，那么将忽略此字段。

PutMsgRecPtr (MQPTR)

这是第一个 MQPMR 放置消息记录的地址。仅当将消息放入分发列表时，才使用 *PutMsgRecPtr*。如果 *RecsPresent* 为零，那么将忽略该字段。

可以使用 *PutMsgRecPtr* 或 *PutMsgRecOffset* 来指定放置消息记录，但不能同时指定两者；请参阅上面 *PutMsgRecOffset* 字段的描述以获取详细信息。如果不使用 *PutMsgRecPtr*，请将其设置为空指针或空字节。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针，否则为全空字节字符串。如果 *Version* 小于 MQPMO_VERSION_2，那么将忽略此字段。

注：在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串，初始值为全空字节字符串。

RecsPresent (MQLONG)

这是应用程序提供的 MQPMR 放入消息记录或 MQRR 响应记录数。仅当将消息放入分发列表时，此数字才能大于零。放置消息记录和响应记录是可选的；应用程序无需提供任何记录，也可以选择仅提供一种类型的记录。但是，如果应用程序提供了这两种类型的记录，那么它必须提供每种类型的 *RecsPresent* 记录。

RecsPresent 的值不必与分发列表中的目标数相同。如果提供的记录过多，那么不会使用多余的记录；如果提供的记录过少，那么会将缺省值用于那些没有放置消息记录的目标的消息属性（请参阅 *PutMsgRecOffset*）。

如果 *RecsPresent* 小于零或大于零但未将消息放入分发列表，那么调用将失败，原因码为 MQRC_RECS_PRESENT_ERROR。

这是一个输入字段。此字段的初始值为 0。如果 *Version* 小于 MQPMO_VERSION_2，那么将忽略此字段。

ResolvedQMgr 名称 (MQCHAR48)

这是本地队列管理器执行名称解析后的目标队列管理器的名称。返回的名称是拥有由 *ResolvedQName* 标识的队列的队列管理器的名称，并且可以是本地队列管理器的名称。

如果 *ResolvedQName* 是本地队列管理器所属的队列共享组所拥有的共享队列，那么 *ResolvedQMgrName* 是队列共享组的名称。如果队列由其他某个队列共享组拥有，那么 *ResolvedQName* 可以是队列共享组的

名称，也可以是作为队列共享组成员的队列管理器的名称 (返回的值的性质由本地队列管理器中存在的队列定义确定)。

仅当对象是单个队列时，才会返回非空白值; 如果对象是分发列表或主题，那么返回的值未定义。

这是输出字段。此字段的长度由 `MQ_Q_MGR_NAME_LENGTH` 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ResolvedQName (MQCHAR48)

这是本地队列管理器执行名称解析后目标队列的名称。返回的名称是由 *ResolvedQMgrName* 标识的队列管理器上存在的队列的名称。

仅当对象是单个队列时，才会返回非空白值; 如果对象是分发列表或主题，那么返回的值未定义。

这是输出字段。此字段的长度由 `MQ_Q_NAME_LENGTH` 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ResponseRec 偏移量 (MQLONG)

这是从 MQPMO 结构开始的第一个 MQRR 响应记录的偏移量 (以字节为单位)。偏移可以是正数或负数。仅当将消息放入分发列表时，才会使用 *ResponseRecOffset*。如果 *RecsPresent* 为零，那么将忽略该字段。

将消息放入分发列表时，可以提供一个或多个 MQRR 响应记录的数组，以标识未成功将消息发送到队列 (MQRR 中的 *CompCode* 字段) 以及每个失败的原因 (MQRR 中的 *Reason* 字段)。可能由于队列未能打开或由于 put 操作失败而未发送消息。仅当调用结果混合时，队列管理器才会设置响应记录 (即，某些消息已成功发送，而其他消息失败，或者所有消息都失败，但原因不同); 来自调用的原因码 `MQRC_MULTIPLE_REASON` 指示此情况。如果相同原因码适用于所有队列，那么将在 MQPUT 或 MQPUT1 调用的 *Reason* 参数中返回该原因，并且不会设置响应记录。

通常，当打开分发列表时，响应记录的数量与 MQOD 指定的对象记录的数量相同; 必要时，每个响应记录都设置为放入相应对象记录所标识的队列的完成代码和原因码。分发列表中未能打开的队列仍必须在阵列中的相应位置为它们分配响应记录，尽管它们设置为由打开操作 (而不是放置操作) 生成的完成代码和原因码。

响应记录数可能与对象记录数不同。如果响应记录少于对象记录，那么应用程序可能无法识别放置操作失败的所有目标或失败原因。如果有比对象记录更多的响应记录，那么不会使用多余的响应记录 (尽管仍必须能够访问这些响应记录)。响应记录是可选的，但如果提供了这些记录，那么其中必须有 *RecsPresent* 个记录。

通过在 *ResponseRecOffset* 中指定偏移量或在 *ResponseRecPtr* 中指定地址，以与 MQOD 中的对象记录类似的方式提供响应记录; 有关如何执行此操作的详细信息，请参阅第 414 页的『MQOD-对象描述符』中描述的 *ObjectRecOffset* 字段。但是，请仅使用 *ResponseRecOffset* 和 *ResponseRecPtr* 中的一个; 如果两者都非零，那么调用将失败，原因码为 `MQRC_RESPONSE_RECORDS_ERROR`。

对于 MQPUT1 调用，此字段必须为零。这是因为响应信息 (如果请求) 是在对象描述符 MQOD 指定的响应记录中返回的。

这是一个输入字段。此字段的初始值为 0。如果 *Version* 小于 `MQPMO_VERSION_2`，那么将忽略此字段。

ResponseRecPtr (MQPTR)

这是第一个 MQRR 响应记录的地址。仅当将消息放入分发列表时，才会使用 *ResponseRecPtr*。如果 *RecsPresent* 为零，那么将忽略该字段。

使用 *ResponseRecPtr* 或 *ResponseRecOffset* 指定响应记录，但不能同时指定这两者; 请参阅上面 *ResponseRecOffset* 字段的描述以获取详细信息。如果不使用 *ResponseRecPtr*，请将其设置为空指针或空字节。

对于 MQPUT1 调用，此字段必须是空指针或空字节。这是因为响应信息 (如果请求) 是在对象描述符 MQOD 指定的响应记录中返回的。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针，否则为全空字节字符串。如果 *Version* 小于 `MQPMO_VERSION_2`，那么将忽略此字段。

注: 在编程语言不支持指针数据类型的平台上, 此字段声明为适当长度的字节字符串, 初始值为全空字节字符串。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQPMO_STRUC_ID

put-message 选项结构的标识。

对于 C 编程语言, 还定义了常量 MQPMO_STRUC_ID_ARRAY; 此值与 MQPMO_STRUC_ID 相同, 但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQPMO_STRUC_ID。

超时 (MQLONG)

这是保留字段; 其值不重要。此字段的初始值为 -1。

UnknownDest 计数 (MQLONG)

这是当前 MQPUT 或 MQPUT1 调用已成功发送到分发列表中解析为远程队列的队列的消息数。队列管理器在分发列表表中临时保留的消息计数为这些分发列表包含的各个目标的数目。将消息放入不在分发列表中的单个队列时, 也会设置此字段。

这是输出字段。此字段的初始值为 0。如果 *Version* 小于 MQPMO_VERSION_1, 那么不会设置此字段。在 z/OS 上未定义此字段, 因为不支持分发列表。

Version (MQLONG)

结构版本号。

值必须为以下其中一项:

MQPMO_VERSION_1

Version-1 put-message 选项结构。

此版本在所有环境中都受支持。

MQPMO_VERSION_2

Version-2 放置消息选项结构。

此版本在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

MQPMO_VERSION_3

Version-3 put-message 选项结构。

此版本在所有环境中都受支持。

仅在结构的最新版本中存在的字段在字段描述中标识为此类字段。以下常量指定当前版本的版本号:

MQPMO_CURRENT_VERSION

当前版本的 put-message 选项结构。

这始终是一个输入字段。此字段的初始值为 MQPMO_VERSION_1。

MQPMO 的初始值和语言声明

| 表 529: MQPMO 中字段的初始值 | | |
|----------------------|-----------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQPMO_STRUC_ID | 'PMO-' |
| <i>Version</i> | MQPMO_VERSION_1 | 1 |
| <i>Options</i> | MQPMO_NONE | 0 |

表 529: MQPMO 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|--------------------------|-------------|---------|
| <i>Timeout</i> | None | -1 |
| <i>Context</i> | None | 0 |
| <i>KnownDestCount</i> | None | 0 |
| <i>UnknownDestCount</i> | None | 0 |
| <i>InvalidDestCount</i> | None | 0 |
| <i>ResolvedQName</i> | None | 空字符串或空白 |
| <i>ResolvedQMgrName</i> | None | 空字符串或空白 |
| <i>RecsPresent</i> | None | 0 |
| <i>PutMsgRecFields</i> | MQPMRF_NONE | 0 |
| <i>PutMsgRecOffset</i> | None | 0 |
| <i>ResponseRecOffset</i> | None | 0 |
| <i>PutMsgRecPtr</i> | None | 空指针或空字节 |
| <i>ResponseRecPtr</i> | None | 空指针或空字节 |
| <i>OriginalMsgHandle</i> | MQHM_NONE | 0 |
| <i>NewMsgHandle</i> | MQHM_NONE | 0 |
| <i>Action</i> | MQACTP_NEW | 0 |
| <i>PubLevel</i> | None | 9 |

注意:

1. 符号 `\n` 表示单个空白字符。
2. 值 `Null` 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。
3. 在 C 编程语言中，宏变量 `MQPMO_DEFAULT` 包含上面列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQPMO MyPMO = {MQPMO_DEFAULT};
```

C 声明

```
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 声明

```

** 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-KNOWNDSTCOUNT PIC S9(9) BINARY.
** Number of messages sent successfully to remote queues
15 MQPMO-UNKNOWNDSTCOUNT PIC S9(9) BINARY.
** Number of messages that could not be sent
15 MQPMO-INVALIDDSTCOUNT 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-NEWMMSGHANDLE PIC S9(18) BINARY.
** The action being performed
15 MQPMO-ACTION PIC S9(9) BINARY.
** Publish level
15 MQPMO-PUBLEVEL PIC S9(9) BINARY.

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

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_UNKNOWNDSTCOUNT 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)

```

Visual Basic 声明

```

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-放置消息记录

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|------------------------|--------|---------------------------------|
| <i>MsgId</i> | 消息标识 | MsgId |
| <i>CorrelId</i> | 相关标识 | CorrelId |
| <i>GroupId</i> | 组标识 | GroupId |
| <i>Feedback</i> | 反馈或原因码 | 反馈 |
| <i>AccountingToken</i> | 记帐标记 | AccountingToken |

MQPMR 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

用途: 将消息放入分发列表时, 使用 MQPMR 结构为单个目标指定各种消息属性。MQPMR 是 MQPUT 和 MQPUT1 调用的输入/输出结构。

字符集和编码: MQPMR 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 MQ 客户机运行, 那么结构必须采用客户机的字符集和编码。

用法: 通过在 MQPUT 或 MQPUT1 调用上提供这些结构的数组, 可以在分发列表中为每个目标队列指定不同的值。某些字段仅为输入, 而其他字段为输入/输出。

注: 这种结构是不常见的, 因为它没有固定的布局。此结构中的字段是可选的, 每个字段的存在或不存在由 MQPMO 中 *PutMsgRecFields* 字段中的标志指示。存在的字段 **必须按以下顺序出现**:

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

不存在的字段在记录中不占用任何空间。

由于 MQPMR 没有固定布局, 因此在头, COPY 和 INCLUDE 文件中没有为受支持的编程语言提供该布局的定义。应用程序员必须创建包含应用程序所需的字段的声明, 并在 *PutMsgRecFields* 中设置标志以指示存在的字段。

MQPMR 的字段

MQPMR 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

AccountingToken (MQBYTE32)

这是要用于发送到队列的消息的记帐令牌，其名称由 MQOPEN 或 MQPUT1 调用上提供的 MQOR 结构数组中的相应元素指定。它的处理方式与 MQMD 中用于放入单个队列的 *AccountingToken* 字段的处理方式相同。请参阅第 364 页的『MQMD - 消息描述符』中 *AccountingToken* 的描述，以获取有关此字段内容的信息。

如果此字段不存在，那么将使用 MQMD 中的值。

这是一个输入字段。

CorrelId (MQBYTE24)

这是要用于发送到队列的消息的相关标识，其名称由 MQOPEN 或 MQPUT1 调用上提供的 MQOR 结构数组中的相应元素指定。它的处理方式与 MQMD 中用于放入单个队列的 *CorrelId* 字段的处理方式相同。

如果此字段在 MQPMR 记录中不存在，或者 MQPMR 记录少于目标，那么 MQMD 中的值将用于那些没有包含 *CorrelId* 字段的 MQPMR 记录的目标。

如果指定了 MQPMO_NEW_CORREL_ID，那么将生成单个新相关标识并用于分发列表中的所有目标，而不考虑它们是否具有 MQPMR 记录。这与 MQPMO_NEW_MSG_ID 的处理方式不同 (请参阅 *MsgId* 字段)。

这是输入/输出字段。

Feedback (MQLONG)

这是要用于发送到队列的消息的反馈代码，其名称由 MQOPEN 或 MQPUT1 调用上提供的 MQOR 结构数组中的相应元素指定。它的处理方式与 MQMD 中用于放入单个队列的 *Feedback* 字段的处理方式相同。

如果此字段不存在，那么将使用 MQMD 中的值。

这是一个输入字段。

GroupId (MQBYTE24)

GroupId 是要用于发送到队列的消息的组标识，其名称由 MQOPEN 或 MQPUT1 调用上提供的 MQOR 结构数组中的相应元素指定。它的处理方式与 MQMD 中用于放入单个队列的 *GroupId* 字段的处理方式相同。

如果此字段在 MQPMR 记录中不存在，或者 MQPMR 记录少于目标，那么 MQMD 中的值将用于那些没有包含 *GroupId* 字段的 MQPMR 记录的目标。该值按 队列上的物理顺序中的记录进行处理，但存在以下差异：

- *GroupId* 是根据 QMName 和时间戳记创建的。因此，要使 *GroupId* 唯一的保留队列管理器名称也保持唯一。也不要再在队列管理器机器上设置回时钟。
- 在将使用新组标识的情况下，队列管理器会为每个目标生成不同的组标识 (即，没有两个目标具有相同的组标识)。
- 在将使用字段中的值的那些情况下，调用失败，原因码为 MQRC_GROUP_ID_ERROR

这是输入/输出字段。

MsgId (MQBYTE24)

这是要用于发送到队列的消息的消息标识，其名称由 MQOPEN 或 MQPUT1 调用上提供的 MQOR 结构数组中的相应元素指定。它的处理方式与 MQMD 中用于放入单个队列的 *MsgId* 字段的处理方式相同。

如果此字段在 MQPMR 记录中不存在，或者 MQPMR 记录少于目标，那么 MQMD 中的值将用于那些没有包含 *MsgId* 字段的 MQPMR 记录的目标。如果该值为 MQMI_NONE，那么将为每个这些目标生成新的消息标识 (即，其中没有两个目标具有相同的消息标识)。

如果指定了 MQPMO_NEW_MSG_ID，那么将为分发列表中的所有目标生成新的消息标识，而无论它们是否具有 MQPMR 记录。这与 MQPMO_NEW_CORREL_ID 的处理方式不同 (请参阅 *CorrelId* 字段)。

这是输入/输出字段。

MQPMR 的初始值和语言声明

没有为此结构定义初始值，因为头，COPY 和 INCLUDE 文件中没有为受支持的编程语言提供任何结构声明。样本声明显示了在所有字段都是必需的情况下如何声明结构。

C 声明

```
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 声明

```
** 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).
```

PL/I 声明

```
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 */
```

Visual Basic 声明

```
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 - 规则和格式化头

本部分描述了规则和格式化头，其中包含的字段以及这些字段的初始值。

MQRFH 概述

可用性: 所有 WebSphere MQ 系统，以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途: MQRFH 结构定义规则和格式化头的布局。使用此头以 "名称/值" 对的形式发送字符串数据。

格式名: MQFMT_RF_HEADER。

字符集和编码: MQRFH 结构 (包括 *NameValueString*) 中的字段是由 MQRFH 之前的头结构中的 *CodedCharSetId* 和 *Encoding* 字段提供的字符集和编码，或者 MQMD 结构中的那些字段 (如果 MQRFH 位于应用程序消息数据的开头) 提供的编码。

字符集必须是对队列名称中有效的字符具有单字节字符的字符集。

MQRFH 的字段

MQRFH 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

CodedCharSetId (MQLONG)

这指定 *NameValueString* 之后的数据的字符集标识; 它不适用于 MQRFH 结构本身中的字符数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。可以使用以下特殊值:

MQCCSI_INHERIT

遵循 此结构的数据中的字符数据与此结构位于同一字符集中。

队列管理器将消息中发送的结构中的此值更改为结构的实际字符集标识。如果未发生错误, 那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER, 那么无法使用 MQCCSI_INHERIT。

此字段的初始值为 MQCCSI_UNDEFINED。

Encoding (MQLONG)

这指定 *NameValueString* 后面的数据的数字编码; 它不适用于 MQRFH 结构本身中的数字数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。

此字段的初始值为 MQENC_NATIVE。

Flags (MQLONG)

可以指定以下内容:

MQRFH_NONE

没有标志。

此字段的初始值为 MQRFH_NONE。

Format (MQCHAR8)

这将指定 *NameValueString* 后面的数据的格式名。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。此字段的编码规则与 MQMD 中 *Format* 字段的编码规则相同。

此字段的初始值为 MQFMT_NONE。

NameValue 字符串 (MQCHARn)

这是包含以下格式的名称/值对的可变长度字符串:

```
name1 value1 name2 value2 name3 value3 ...
```

每个名称或值必须与相邻的名称或值用一个或多个空白字符分隔; 这些空白不重要。名称或值可以包含重要空格, 方法是使用双引号对名称或值添加前缀和后缀; 将打开的双引号和匹配的右双引号之间的所有字符视为重要字符。在以下示例中, 名称为 FAMOUS_WORDS, 值为 Hello World:

```
FAMOUS_WORDS "Hello World"
```

名称或值可以包含除空字符以外的任何字符 (充当 *NameValueString* 的定界符)。但是, 为了帮助实现互操作性, 应用程序可以将名称限制为以下字符:

- 第一个字符: 大写或小写字母 (A 到 Z 或 a 到 z) 或下划线。
- 后续字符: 大写或小写字母, 十进制数字 (0 到 9), 下划线, 连字符或点。

如果名称或值包含一个或多个双引号, 那么名称或值必须用双引号括起来, 并且字符串中的每个双引号都必须加倍:

```
Famous_Words "The program displayed ""Hello World"""
```

名称和值区分大小写，即，不会将小写字母视为与大写字母相同。例如，FAMOUS_WORDS 和 Famous_Words 是两个不同的名称。

NameValueString 的长度 (以字节计) 等于 *StrucLength* 减去 MQRFH_STRUC_LENGTH_FIXED。为了避免在某些环境中转换用户数据时出现问题，请使此长度为 4 的倍数。将 *NameValueString* 用空格填充到此长度，或者通过在字符串中的最后一个有效字符后面放置空字符来提前将其终止。将忽略空字符及其后面的字节 (最多为指定长度 *NameValueString*)。

注: 由于此字段的长度不固定，因此将从为受支持的编程语言提供的结构声明中省略此字段。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQRFH_STRUC_ID

规则和格式化头结构的标识。

对于 C 编程语言，还定义了常量 MQRFH_STRUC_ID_ARRAY; 此值与 MQRFH_STRUC_ID 相同，但是字符数组而不是字符串。

此字段的初始值为 MQRFH_STRUC_ID。

StrucLength (MQLONG)

这是 MQRFH 结构的长度 (以字节计)，包括结构末尾的 *NameValueString* 字段。长度不包含任何跟在 *NameValueString* 字段后面的用户数据。

为避免在某些环境中转换用户数据时发生问题，*StrucLength* 必须是四的倍数。

以下常量给出结构的 *fixed* 部分的长度，即不包括 *NameValueString* 字段的长度:

MQRFH_STRUC_LENGTH_FIXED

MQRFH 结构的固定部分的长度。

此字段的初始值为 MQRFH_STRUC_LENGTH_FIXED。

Version (MQLONG)

这是结构版本号; 值必须为:

MQRFH_VERSION_1

Version-1 规则和格式化头结构。

此字段的初始值为 MQRFH_VERSION_1。

MQRFH 的初始值和语言声明

| 表 531: MQRFH 的 MQRFH 中字段的初始值 | | |
|------------------------------|--------------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQRFH_STRUC_ID | 'RFH↵' |
| <i>Version</i> | MQRFH_VERSION_1 | 1 |
| <i>StrucLength</i> | MQRFH_STRUC_LENGTH_FIXED | 32 |
| <i>Encoding</i> | MQENC_NATIVE | 取决于环境 |
| <i>CodedCharSetId</i> | MQCCSI_UNDEFINED | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQRFH_NONE | 0 |

表 531: MQRFH 的 MQRFH 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|---|-------|-----|
| 注意: | | |
| 1. 符号 <code>\n</code> 表示单个空白字符。 2. 在 C 编程语言中, 宏变量 <code>MQRFH_DEFAULT</code> 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值: | | |
| <pre>MQRFH MyRFH = {MQRFH_DEFAULT};</pre> | | |

C 声明

```
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 声明

```
** 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.
```

PL/I 声明

```
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 */
```

High Level Assembler 声明

| | | | |
|----------------------|-------|------------------|---------------------------------------|
| 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) | |

Visual Basic 声明

```

Type MQRFH
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Total length of MQRFH including'
  Encoding     As Long     'NameValueString'
  CodedCharSetId As Long   'Numeric encoding of data that follows'
  Format       As String*8 'NameValueString'
  Flags       As Long     'Character set identifier of data that'
  End Type

```

MQRFH2 -规则 and 格式化头 2

本部分描述了规则和格式化头 2，它包含哪些字段以及这些字段的初始值。

MQRFH2 的概述

可用性

所有 WebSphere MQ 系统以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途

MQRFH2 头基于 MQRFH 头，但它允许在不进行转换的情况下传输 Unicode 字符串，并且可以包含数字数据类型。

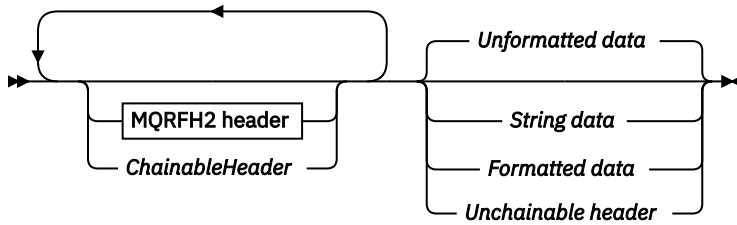
MQRFH2 结构定义 version-2 规则和格式化头的格式。使用此头来发送已使用类似 XML 的语法进行编码的数据。一条消息可以连续包含两个或多个 MQRFH2 结构，用户数据可以选择性地跟在序列中的最后一个 MQRFH2 结构之后。

格式名

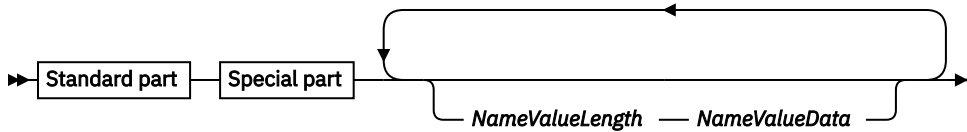
MQFMT_RF_HEADER_2

Syntax

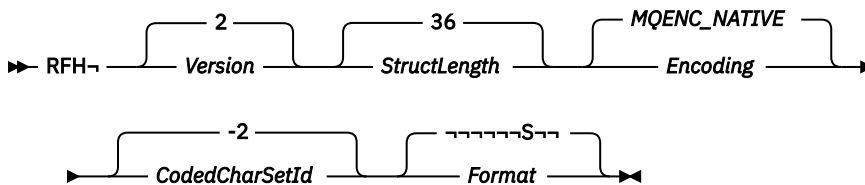
WebSphere MQ Message



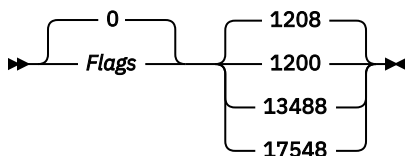
MQRFH2 header



Standard part



Special part



字符集和编码

特殊规则适用于用于 MQRFH2 结构的字符集和编码:

- *NameValueData* 以外的字段位于由 MQRFH2 之前的头结构中的 *CodedCharSetId* 和 *Encoding* 字段提供的字符集和编码中, 或者由 MQMD 结构中的那些字段提供 (如果 MQRFH2 位于应用程序消息数据的开头)。

字符集必须是对队列名称中有效的字符具有单字节字符的字符集。

在 MQGET 调用上指定 MQGMO_CONVERT 时, 队列管理器会将除 *NameValueData* 以外的 MQRFH2 字段转换为请求的字符集和编码。

- *NameValueData* 位于 *NameValueCCSID* 字段提供的字符集中。仅列出的 Unicode 字符集对 *NameValueCCSID* 有效; 请参阅 *NameValueCCSID* 的描述以获取详细信息。

某些字符集具有依赖于编码的表示。如果 *NameValueCCSID* 是其中一个字符集, 那么 *NameValueData* 必须与 MQRFH2 中的其他字段采用相同的编码。

在 MQGET 调用上指定 MQGMO_CONVERT 时, 队列管理器会将 *NameValueData* 转换为请求的编码, 但不会更改其字符集。

MQRFH2 的字段

MQRFH2 结构包含以下字段; 这些字段按字母顺序进行描述:

CodedCharSetId (MQLONG)

这指定最后一个 *NameValueData* 字段后面的数据的字符集标识; 它不适用于 MQRFH2 结构本身中的字符数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。可以使用以下特殊值:

MQCCSI_INHERIT

遵循 此结构的数据中的字符数据与此结构位于同一字符集中。

队列管理器将消息中发送的结构中的此值更改为结构的实际字符集标识。如果未发生错误, 那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER, 那么无法使用 MQCCSI_INHERIT。

此字段的初始值为 MQCCSI_INHERIT。

Encoding (MQLONG)

这将指定最后一个 *NameValueData* 字段后面的数据的数字编码; 它不适用于 MQRFH2 结构本身中的数字数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。

此字段的初始值为 MQENC_NATIVE。

Flags (MQLONG)

此字段的初始值为 MQRFH_NONE。MQRFH_NONE 必须指定。

MQRFH_NONE

没有标志。

MQRFH_INTERNAL

MQRFH2 头包含内部设置的属性。

MQRFH_INTERNAL 供队列管理器使用。

前 16 位 MQRFH_FLAGS_RESTRICTED_MASK 保留用于队列管理器设置的标志。用户可能设置的标志在底部 16 位中定义。

Format (MQCHAR8)

这将指定最后一个 *NameValueData* 字段后面的数据的格式名称。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。此字段的编码规则与 MQMD 中 *Format* 字段的编码规则相同。

此字段的初始值为 MQFMT_NONE。

NameValueCCSID (MQLONG)

这将在 *NameValueData* 字段中指定数据的编码字符集标识。这与 MQRFH2 结构中其他字符串的字符集不同, 并且可以与结构末尾最后一个 *NameValueData* 字段后面的数据 (如果有) 的字符集不同。

NameValueCCSID 必须具有下列其中一个值:

| CCSID | 含义 |
|--------------|-----------------------|
| 1200 | UCS-2 开放式 |
| 13488 | UCS-2 2.0 子集 |
| 17584 | UCS-2 2.1 子集 (包含欧元符号) |
| 1208 | UTF-8 |

对于 UCS-2 字符集, *NameValueData* 的编码 (字节顺序) 必须与 MQRFH2 结构中其他字段的编码相同。不支持替代字符 (X'D800'到 X'DFFF')。

注: 如果 *NameValueCCSID* 没有上面列出的某个值, 并且 MQRFH2 结构需要在 MQGET 调用上进行转换, 那么调用将完成, 原因码为 MQRC_SOURCE_CCSD_ERROR, 并且将返回未转换的消息。

此字段的初始值为 1208。

NameValue 数据 (MQCHARn)

NameValueData 是一个可变长度字段，其中包含包含消息属性的名称/值对的文件夹。文件夹是包含使用类似 XML 的语法编码的数据的可变长度字符串。字符串的长度 (以字节为单位) 由 *NameValueLength* 字段之前的 *NameValueLength* 字段给出。长度必须是 4 的倍数。

NameValueLength 和 *NameValueData* 字段是可选的，但如果存在，那么它们必须作为一对且相邻。字段对可以根据需要重复多次，例如：

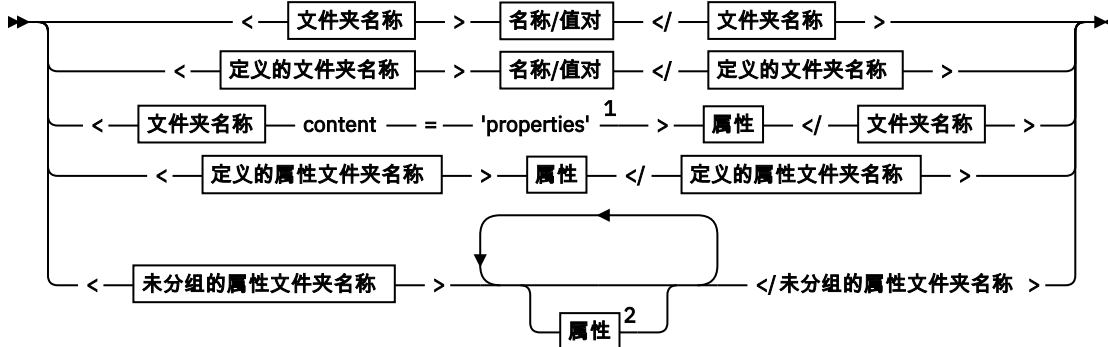
```
length1 data1 length2 data2 length3 data3
```

NameValueData 未转换为 MQGET 调用上指定的字符集。即使使用生效的 MQGMO_CONVERT 选项检索消息，*NameValueData* 也会保留在其原始字符集中。但是，*NameValueData* 将转换为 MQGET 调用上指定的编码。

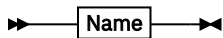
注：由于这些字段是可选的，因此将从为支持的各种编程语言提供的结构声明中省略这些字段。

注：语法图中使用了“定义的”和“保留的”术语。“已定义”表示名称由 IBM WebSphere MQ 使用。“保留”表示保留该名称以供 WebSphere MQ 将来使用。

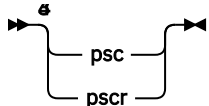
NameValueData 语法



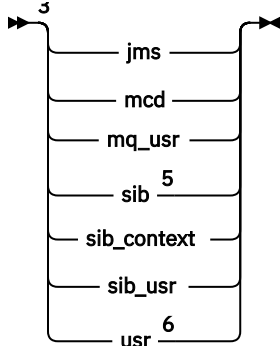
文件夹名称



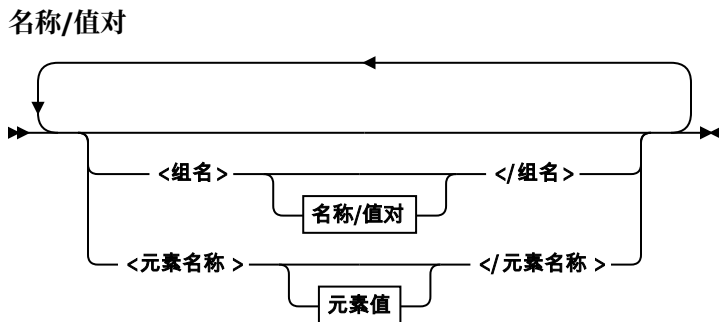
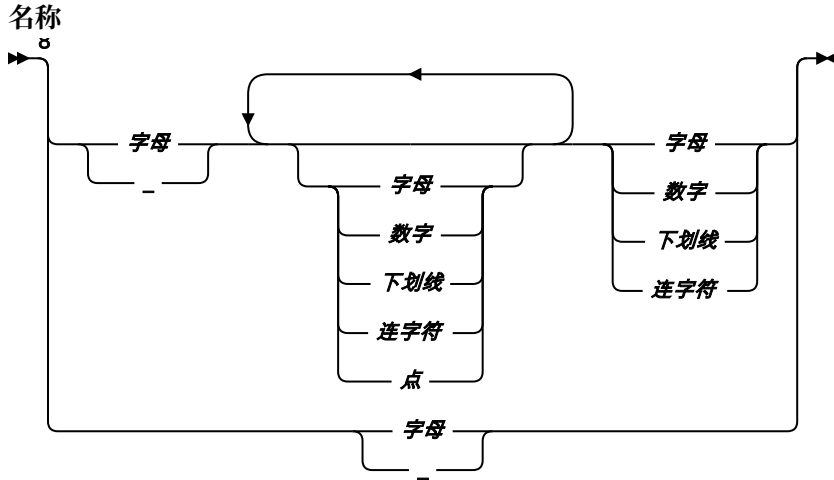
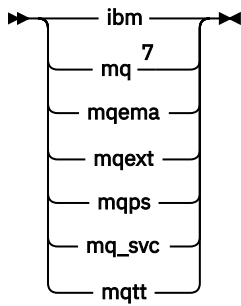
定义的文件夹名称



定义的属性文件夹名称



未分组的属性文件夹名称



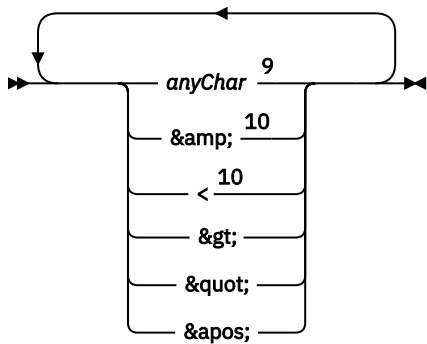
组名



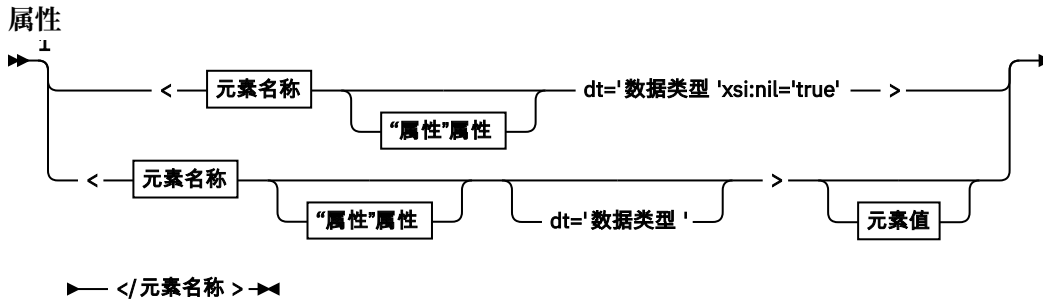
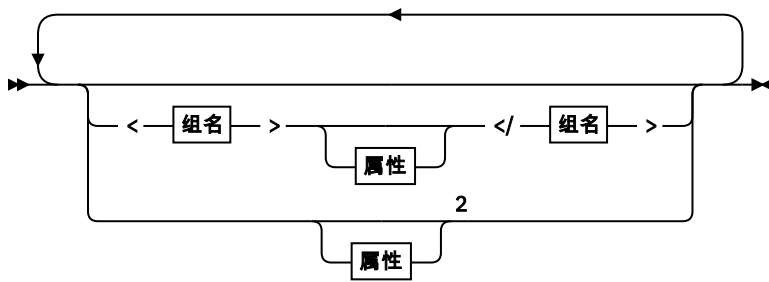
元素名称



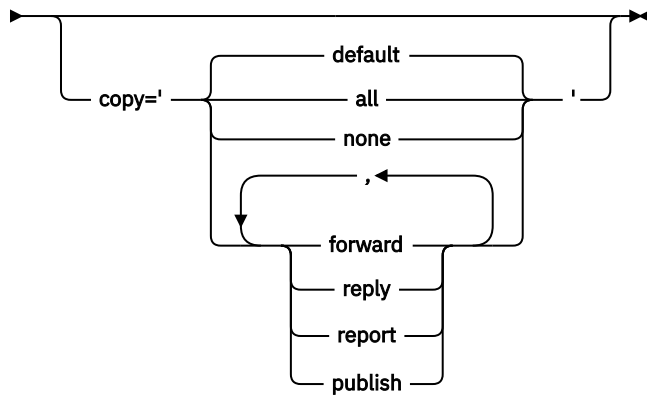
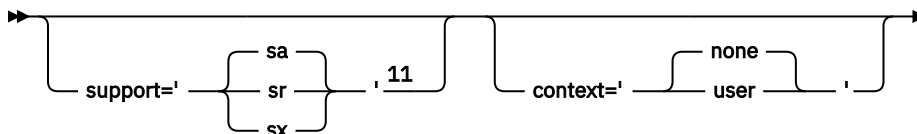
元素值



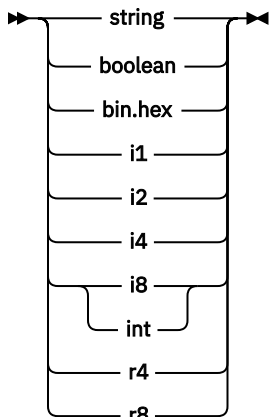
属性



“属性”属性



数据类型



注：

¹ 双引号或单引号有效。

- 2 请勿使用无效属性名; 请参阅 第 472 页的『属性名无效』。 仅将保留属性名称用于其定义的用途; 请参阅 第 472 页的『定义的属性名』。
- 3 名称必须为小写。
- 4 仅支持一个 psc 和 pscr 文件夹。
- 5 只有第一个 MQRFH2 头中的属性很重要。 WebSphere Application Server Service Integration Bus 将忽略后续 MQRFH2 头中的 sib, sib_context 和 sib_usr 文件夹。
- 6 MQRFH2 中不得存在多个 usr 文件夹。 usr 文件夹中的属性只能出现一次。
- 7 只有第一个 mq 文件夹中的属性才有意义。 如果文件夹为 UTF-8, 那么仅支持单字节 UTF-8 字符。 唯一的空格字符是 Unicode U+0020。
- 8 有效字符在 W3C XML 规范中定义, 基本上由 Unicode 类别 Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm, 和 Nd 组成。
- 9 所有字符都很重要。 前导和尾部空格是元素值的一部分。
- 10 请勿使用无效字符; 请参阅 第 472 页的『无效字符』。 使用转义序列, 而不是这些无效字符。
- 11 支持属性属性仅对 mq 文件夹有效

文件夹名称

NameValueData 包含单个文件夹。 要创建多个文件夹, 请创建多个 *NameValueData* 字段。 您可以在消息中的单个 MQRFH2 头中创建多个 *NameValueData* 字段。 或者, 您可以创建多个链式 MQRFH2 头, 每个头包含多个 *NameValueData* 字段。

MQRFH2 头的顺序和 *NameValueData* 字段的顺序与文件夹的逻辑内容没有差别。 如果同一文件夹在消息中存在多次, 那么会将该文件夹作为一个整体进行解析。 如果同一属性出现在同一文件夹的多个实例中, 那么会将其解析为列表。

MQRFH2 的正确解析不受可物理存储在消息中的文件夹的替代方法的影响。

四个文件夹不遵循此规则。 仅解析 mq, sib, sib_context 和 sib_usr 文件夹的第一个实例。

如果同一属性在链式 MQRFH2 头的组合内容中多次出现, 那么将仅解析该属性的第一个实例。 如果使用 API 调用 (例如 MQSETMP) 设置属性并由应用程序直接添加到 MQRFH2, 那么 API 调用优先。

文件夹名称是包含名称/值对或组的文件夹的名称。 可以在文件夹树中的同一级别混合组和名称/值对; 请参阅 第 463 页的图 1。 请勿组合组名和元素名称; 请参阅 第 463 页的图 2

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>  
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

图 1: 正确使用组和名称/值对

```
<group1><nvp1>value</nvp1>value</group1>
```

图 2: 组和名称/值对的使用不正确

请勿使用无效或保留的文件夹名称; 请参阅 第 472 页的『无效的路径名』 和 第 472 页的『保留文件夹或属性文件夹名称』。 仅将定义的文件夹名称用于其定义的用途; 请参阅 第 464 页的『定义的文件夹名称』。

如果将属性 'content=properties' 添加到文件夹名称标记, 那么该文件夹将成为属性文件夹; 请参阅 第 463 页的图 3。

```
<myFolder></myfolder>  
<myPropertyFolder contents='properties'></myPropertyFolder>
```

图 3: 文件夹和属性文件夹的示例

文件夹名称区分大小写。文件夹名称和属性文件夹名称共享同一名称空间。它们必须具有不同的名称。第 464 页的图 4 中的 Folder1 必须是与第 464 页的图 5 中的 Folder2 不同的名称。

```
<Folder1><NVP1>value</NVP1></Folder1>
```

图 4: Folder1 名称空间

```
<Folder2 content='properties'><Property1>value</Property1></Folder2>
```

图 5: Folder2 名称空间

不同文件夹中的组，属性和名称/值对具有不同的名称空间。第 464 页的图 5 中的 Property1 是与第 464 页的图 6 中的 Property1 不同的属性。

```
<Folder3 content='properties'><Property1>value</Property1></Folder3>
```

图 6: Folder3 名称空间

属性文件夹在两个重要方面与非属性文件夹不同：

1. 属性文件夹包含属性，非属性文件夹包含名称/值对。这些文件夹在语法上略有不同。
2. 使用定义的接口（例如，属性 MQI 或 JMS 消息属性）来访问消息属性。这些接口确保 MQRFH2 中的属性文件夹格式正确。格式良好的属性文件夹在不同平台和不同发行版上的队列管理器之间可互操作。

消息属性 MQI 是读取和写入 MQRFH2 的稳健方法，可避免正确解析 MQRFH2 的困难。

定义的文件夹名称

定义的文件夹名称是保留供 WebSphere MQ 或其他产品使用的文件夹的名称。请勿创建同名文件夹，也不要将您自己的名称/值对添加到文件夹中。定义的文件夹为 psc 和 pscr。

psc 和 pscr 由已排队的发布/预订使用。

与 MQMF_SEGMENT 或 MQMF_SEGMENTATION_ALLOWED 一起放入的分段消息不能包含具有已定义文件夹名称的 MQRFH2。MQPUT 失败，原因码为 2443，MQRC_SEGMENTATION_NOT_ALLOWED。

定义的属性文件夹名称

定义的属性文件夹名称是 IBM WebSphere MQ 或其他产品所使用的属性文件夹的名称。有关文件夹的名称及其内容，请参阅 [属性文件夹](#)。定义的属性文件夹名称是 WebSphere MQ 保留的所有文件夹名称的子集；请参阅第 472 页的『[保留文件夹或属性文件夹名称](#)』。

存储在已定义属性文件夹中的任何元素都是属性。存储在已定义属性文件夹中的元素不得具有 content='properties' 属性。

只能将属性添加到定义的属性文件夹 usr，mq_usr 和 sib_usr。在其他属性文件夹（例如 mq 和 sib）中，WebSphere MQ 将忽略或丢弃它无法识别的属性。

每个已定义的属性文件夹的描述都列出了 IBM WebSphere MQ 已定义的可供应用程序使用的属性。某些属性是通过设置或获取 JMS 属性间接访问的，而某些属性是使用 MQSETMP 和 MQINQMP MQI 调用直接访问的。

定义的属性文件夹还包含 IBM WebSphere MQ 已保留但应用程序无权访问的其他属性。未列出保留属性的名称。usr，mq_usr 和 sib_usr 属性文件夹中不存在保留属性。但是，请勿创建具有无效属性名的属性；请参阅第 472 页的『[属性名无效](#)』。

属性文件夹

jms

jms 包含 JMS 头字段，以及无法在 MQMD 中完全表达的 JMSX 属性。jms 文件夹始终存在于 JMS MQRFH2 中。

| 属性同义词 | 属性名 | 数据类型 | 文件夹 |
|----------------|---------|--------|---|
| 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> |
| JMSXGroupID | jms.Gid | string | <jms><Gid>groupId</Gid></jms> |
| JMSXGroupSeq | jms.Seq | i4 | <jms><Seq>messageSequenceNo</Seq></jms> |

不要在 jms 文件夹中添加您自己的属性。

mcd

mcd 包含描述消息的格式的属性。例如，消息服务域 Msd 属性将 JMS 消息标识为 JMSTextMessage、JMSBytesMessage、JMSStreamMessage、JMSMapMessage、JMSObjectMessage 或 null。

mcd 文件夹始终出现在包含 MQRFH2 的 JMS 消息中。

它始终存在于包含从 WebSphere Message Broker 发送的 MQRFH2 的消息中。它描述消息的域、格式、类型和消息集。

| 属性同义词 | 属性名 | 数据类型 | 文件夹 |
|-------|----------|--------|---------------------------------------|
| | 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> |

不要在 mcd 文件夹中添加您自己的属性。

mq_usr

mq_usr 包含未公开为 JMS 用户定义属性的应用程序定义属性。不满足 JMS 需求的属性可以放置在此文件夹中。

您可以在 mq_usr 文件夹中创建属性。您在 mq_usr 中创建的属性类似于您在具有 content='properties' 属性的新文件夹中创建的属性。

sib

sib 包含 WebSphere Application Server 服务集成总线 (WAS/SIB) 系统消息属性。sib 属性不会作为 JMS 属性向 IBM WebSphere MQ JMS 应用程序公开，因为它们不是受支持的类型。例如，某些 sib 属性无法公开为 JMS 属性，因为它们是字节数组。某些 sib 属性作为 JMS_IBM_* 属性向 WAS/SIB 应用程序公开；这些属性包括正向和反向路由路径属性。

不要在 sib 文件夹中添加您自己的属性。

sib_context

sib_context 包含未向 WAS/SIB 用户应用程序公开或未作为 JMS 属性公开的 WAS/SIB 系统消息属性。sib_context 包含用于 Web Service 的安全性和事务性属性。

不要在 sib_context 文件夹中添加您自己的属性。

sib_usr

sib_usr 包含未公开为 JMS 用户属性的 WAS/SIB 用户消息属性，因为它们不是受支持的类型。sib_usr 在 SIMessage 界面中向 WAS/SIB 应用程序公开；请参阅 [开发服务集成](#)。

sib_usr 属性的类型必须为 bin.hex，并且值的格式必须正确。如果 IBM WebSphere MQ 应用程序将 bin.hex 类型的元素以错误格式写入文件夹，那么应用程序将接收到 IOException。如果该属性的数据类型不是 bin.hex，那么应用程序将接收到 ClassCastException。

请勿尝试使用此文件夹使 JMS 用户属性可供 WAS/SIB 使用；而是使用 usr 文件夹。

您可以在 sib_usr 文件夹中创建属性。

usr

usr 包含与消息关联的应用程序定义 JMS 属性。仅当应用程序已设置应用程序定义的属性时，usr 文件夹才存在。

usr 是缺省属性文件夹。如果在没有文件夹名称的情况下设置了属性，那么会将其放置在 usr 文件夹中。

| 属性同义词 | 属性名 | 数据类型 | 文件夹 |
|-------|----------------------|--------|---|
| | 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> |

您可以在 `usr` 文件夹中创建属性。

使用 `MQMF_SEGMENT` 或 `MQMF_SEGMENTATION_ALLOWED` 放入的分段消息不能包含具有已定义属性文件夹名称的 `MQRFH2`。MQPUT 失败，原因码为 2443，`MQRC_SEGMENTATION_NOT_ALLOWED`。

未分组的属性文件夹名称

ibm

`ibm` 包含仅由 IBM WebSphere MQ 使用的属性。

| 属性同义词 | 属性名 | 数据类型 | 文件夹 |
|-------|----------------------|--------|--|
| | <code>ibm.rfp</code> | string | <code><ibm><rfp>fingerprint</rfp></ibm></code> |

不要在 `ibm` 文件夹中添加您自己的属性。

mq

`mq` 包含仅由 IBM WebSphere MQ 使用的属性。

以下限制适用于 `mq` 文件夹中的属性：

- MQ 仅对消息中第一个重要 `mq` 文件夹中的属性执行操作；将忽略消息中任何其他 `mq` 文件夹中的属性。
- 文件夹中只允许使用单字节 UTF-8 字符。文件夹中的多字节字符可能导致解析失败，并且要拒绝消息。
- 请勿在文件夹中使用转义字符串。转义字符串被视为元素的实际值。
- 只有 Unicode 字符 `U+0020` 被视为文件夹中的空格。所有其他字符都被视为重要字符，并可能导致文件夹解析失败以及要拒绝的消息。

如果 `mq` 文件夹的解析失败，或者该文件夹未遵守这些限制，那么将拒绝此消息，原因码为 2527，`MQRC_RFH_RESTRICTED_FORMAT_ERR`。

不要在 `mq` 文件夹中添加您自己的属性。

mqema

`mqema` 包含仅供 WebSphere Application Server 使用的属性。文件夹已替换为 `mqext`。

不要在 `mqema` 文件夹中添加您自己的属性。

mqext

`mqext` 包含仅供 WebSphere Application Server 使用的属性。仅当应用程序至少设置了一个 IBM 定义属性时，才会显示该文件夹。

| 属性同义词 | 属性名 | 数据类型 | 文件夹 |
|-------------------------------|------------------------|--------|--|
| <code>JMSArmCorrelator</code> | <code>mqext.Arm</code> | string | <code><mqext><Arm>armCorrelator</Arm></mqext></code> |
| <code>JMSRMCorrelator</code> | <code>mqext.Wrm</code> | string | <code><mqext><Wrm>wrmCorrelator</Wrm></mqext></code> |

不要在 `mqext` 文件夹中添加您自己的属性。

mqps

mqps 包含仅由 IBM WebSphere MQ 发布/预订使用的属性。 仅当应用程序已设置至少一个集成的发布/预订属性时，该文件夹才存在。

| 属性同义词 | 属性名 | 数据类型 | 文件夹 |
|-----------------|------------|---------|---|
| 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> |
| MQPubStrInpData | mqpse.Sid | string | <mqps><Sid>publicationData</Sid></mqps> |
| MQPubFormat | mqpse.Pfmt | i8 | <mqps><Pfmt>messageFormat</Pfmt></mqps> |

不要在 mqps 文件夹中添加您自己的属性。

mq_svc

mq_svc 包含 SupportPac MA93 所使用的属性。

不要在 mq_svc 文件夹中添加您自己的属性。

mqtt

mqtt 包含 IBM WebSphere MQ Telemetry 使用的属性

| 属性同义词 | 属性名 | 数据类型 | 文件夹 |
|-------|---------------|--------|---|
| | mqtt.clientId | string | <mqtt><clientId>topicString</clientId></mqtt> |
| | mqtt.qos | i4 | <mqtt><qos>qualityOfService</qos></mqtt> |
| | mqtt.msgid | string | <mqtt><msgid>messageIdentifier</msgid></mqtt> |

不要在 mqtt 文件夹中添加您自己的属性。

与 MQMF_SEGMENT 或 MQMF_SEGMENTATION_ALLOWED 一起放入的分段消息不能包含具有未分组的属性文件夹名称的 MQRFH2。MQPUT 失败，原因码为 2443，MQRC_SEGMENTATION_NOT_ALLOWED。

名称/值对

在语法图中，“名称/值对”描述普通文件夹的内容。普通文件夹包含组和元素。元素是名称/值对。一个组包含元素和其他组。

就树而言，元素是叶节点，组是内部节点。内部节点和文件夹（即根节点）可以包含内部节点和叶节点的混合。节点不能同时是内部节点和叶节点；请参阅 [第 463 页的图 2](#)。

属性

在语法图中，“属性”描述了属性文件夹的内容。属性文件夹包含组和属性。属性是具有可选数据类型属性的名称/值对。组包含属性和其他组。

就树而言，属性是叶节点，组是内部节点。内部节点和属性文件夹（即根节点）可以同时包含内部节点和叶节点。节点不能同时是内部节点和叶节点；请参阅 [第 463 页的图 2](#)。

属性

消息属性是属性文件夹中的名称/值对。它可以选择包含数据类型属性和属性属性；有关示例，请参阅 [第 469 页的图 7](#)。如果省略了数据类型属性，那么属性类型为 `string`。

```
<pf><p1 dt='i8' >value</p1></pf>
```

图 7: 数据类型属性

消息属性的名称是其完整路径名，使用类似 XML 的 `<>` 语法替换为点。例如，`myPropertyFolder1.myGroup1.myGroup2.myProperty1` 映射到 [第 469 页的图 8](#) 中的 `NameValueData` 字符串。将对字符串进行格式化，以便更轻松地进行读取。

```
<myPropertyFolder1>
  <myGroup1>
    <myGroup2>
      <myProperty1>value</myProperty1>
    </myGroup2>
  </myGroup1>
</myPropertyFolder1>
```

图 8: 单个属性名称映射

属性文件夹可以包含多个属性。例如，[第 469 页的图 9](#) 中的属性映射到 [第 470 页的图 10](#) 中的属性文件夹

```
myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

图 9: 具有相同根名称的多个属性

```

<myPropertyFolder1>
  <myProperty4>value</myProperty4>
  <myGroup1>
    <myGroup2>
      <myProperty1>value</myProperty1>
      <myProperty2>value</myProperty2>
    </myGroup2>
    <myProperty3>value</myProperty3>
  </myGroup1>
</myPropertyFolder1>

```

图 10: 多个属性名称映射

名称

名称必须以字母或下划线开头。它不得包含 *Colon*，不得以句点结束，并且只能包含字母，数字，下划线，连字符和 *Dots*。有效字符在 W3C XML 规范中定义，基本上由 Unicode 类别 L1, Lu, Lo, Lt, N1, Mc, Mn, Lm, 和 Nd 组成。

属性或名称/值对的完整路径不得违反第 472 页的『无效的路径名』中描述的规则。路径限制为 4095 字节，不得包含 Unicode 兼容性字符，也不得以字符串 XML 开头。

组名

组名的语法与名称相同。组名是可选的。属性和名称/值对可以放在文件夹的根目录中。如果组有助于组织属性和名称/值对，请使用组。

元素名称

元素名称的语法与名称相同。

元素值

元素值包含 `<Element name>` 标记与 `</Element name>` 之间的所有空格。请勿在值中使用两个字符 `<` 和 `&`。然后替换为 `<` 和 `&`。

“属性”属性

属性属性映射属性描述符字段: 映射如下所示:

支持

```

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

请勿将 all 与其他选项结合使用。

default

MQPD_COPY_DEFAULT

请勿将 default 与其他选项结合使用。default 与 forward + report + publish 相同

none

MQPD_COPY_NONE

请勿将 none 与其他选项结合使用。

支持 属性属性仅适用于 mq 文件夹中的属性。

上下文 和 CopyOptions 属性属性适用于所有属性文件夹。

数据类型

MQRFH2 数据类型映射到消息属性类型，如下所示：

| 表 539: 数据类型映射 | |
|---------------|-----------|
| MQRFH2 数据类型 | 消息属性类型 |
| bin.hex | MQBYTE [] |
| boolean | MQBOOL |
| i1 | MQINT8 |
| i2 | MQINT16 |
| i4 | MQINT32 |
| i8 | MQINT64 |
| r4 | MQFLOAT32 |
| r8 | MQFLOAT64 |
| string | MQCHAR [] |

假定没有数据类型的任何元素的类型为 string。

元素属性 xsi:nil='true' 指示空值。请勿将属性 xsi:nil='false' 用于非空值。例如，以下属性具有空值：

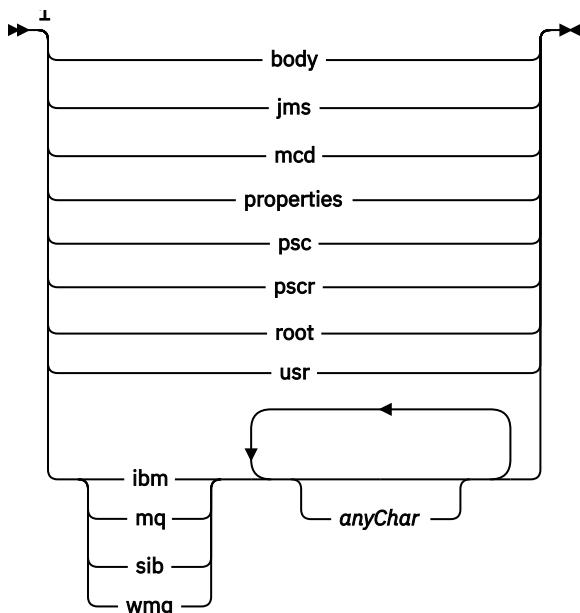
```
<NullProperty
xsi:nil='true'></NullProperty>
```

字节或字符串属性可以具有空值。空值由具有零长度元素值的 MQRFH2 元素表示。例如，以下属性具有空值：

```
<EmptyProperty></EmptyProperty>
```

保留文件夹或属性文件夹名称

将文件夹或属性文件夹的名称限制为不以以下任何字符串开头。前缀是为 IBM 创建的文件夹或属性名称保留的。

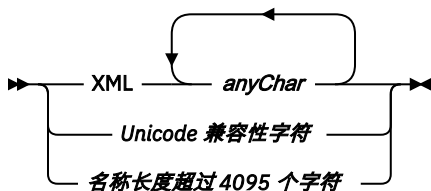


注：

¹ 保留文件夹或属性名称包含小写字母和大写字母的任意组合。

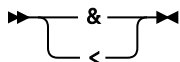
无效的路径名

将名称/值对或属性的完整路径限制为不包含以下任何字符串。



无效字符

始终使用转义序列 & 和 <, 而不是文字 “&” 和 “<”。

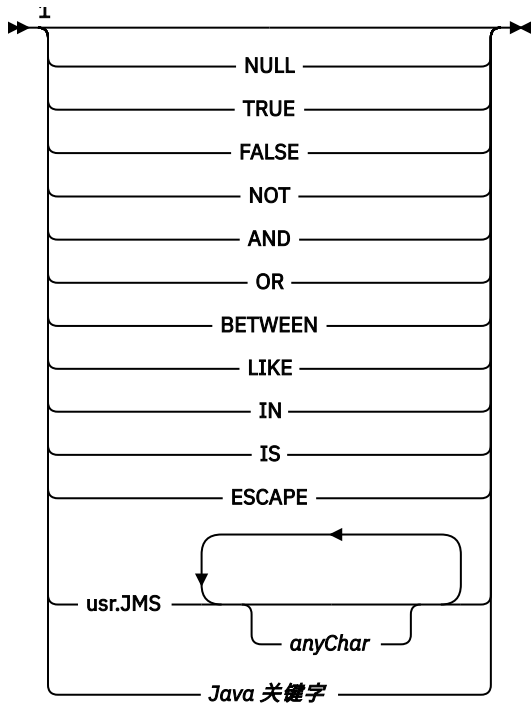


定义的属性名

定义的属性名称是由 WebSphere MQ 或其他产品定义并由 IBM WebSphere MQ 和用户应用程序使用的属性的名称。定义的属性仅存在于定义的属性文件夹中。在属性文件夹的描述中描述了定义的属性名称; 请参阅属性文件夹。

属性名无效

请勿构造与以下规则匹配的属性名。此规则适用于命名属性的完整属性路径, 而不仅仅适用于属性元素名称。



注:

¹ 无效的属性名称可以包含大写和小写的任意组合。

NameValue 长度 (MQLONG)

相应 NameValueData 字段的长度

这将在 NameValueData 字段中指定数据的长度 (以字节计)。NameValueLength 必须是四的倍数。

注: NameValueLength 和 NameValueData 字段是可选的, 但如果存在, 那么它们必须作为一对且相邻。字段对可以根据需要重复多次, 例如:

```
length1 data1 length2 data2 length3 data3
```

由于这些字段是可选的, 因此将从为支持的各种编程语言提供的结构声明中省略这些字段。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQRFH_STRUC_ID

规则和格式化头结构的标识。

对于 C 编程语言, 还定义了常量 MQRFH_STRUC_ID_ARRAY; 此值与 MQRFH_STRUC_ID 相同, 但是字符数组而不是字符串。

此字段的初始值为 MQRFH_STRUC_ID。

StrucLength (MQLONG)

这是 MQRFH2 结构的长度 (以字节计), 包括结构末尾的 NameValueLength 和 NameValueData 字段。对于结构末尾的多对 NameValueLength 和 NameValueData 字段有效, 顺序如下:

```
length1, data1, length2, data2, ...
```

StrucLength 不包含可能跟在结构末尾的最后一个 NameValueData 字段之后的任何用户数据。

为避免在某些环境中转换用户数据时出现问题, StrucLength 必须是四的倍数。

以下常量给出了结构的 fixed 部分的长度, 即不包括 NameValueLength 和 NameValueData 字段的长度:

MQRFH_STRUC_LENGTH_FIXED_2

MQRFH2 结构的固定部分的长度。

此字段的初始值为 MQRFH_STRUC_LENGTH_FIXED_2。

Version (MQLONG)

这是结构版本号; 值必须为:

MQRFH_VERSION_2

Version-2 规则和格式化头结构。

此字段的初始值为 MQRFH_VERSION_2。

MQRFH2 的初始值和语言声明

| 表 540: MQRFH2 的 MQRFH2 中字段的初始值 | | |
|--------------------------------|----------------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQRFH_STRUC_ID | 'RFH↵' |
| <i>Version</i> | MQRFH_VERSION_2 | 2 |
| <i>StrucLength</i> | MQRFH_STRUC_LENGTH_FIXED_2 | 36 |
| <i>Encoding</i> | MQENC_NATIVE | 取决于环境 |
| <i>CodedCharSetId</i> | MQCCSI_INHERIT | -2 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQRFH_NONE | 0 |
| <i>NameValueCCSID</i> | None | 1208 |

注意:

1. 符号 ↵ 表示单个空白字符。
2. 在 C 编程语言中, 宏变量 MQRFH2_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};
```

C 声明

```
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 声明

```

**  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.

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

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
MQRFH_AREA     DS   CL(MQRFH_LENGTH)

```

Visual Basic 声明

```

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'

```

```

Format          As String*8  'follows last NameValueData field'
                'Format name of data that follows last'
                'NameValueData field'
Flags           As Long   'Flags'
NameValueCCSID As Long   'Character set identifier of NameValueData'
End Type

```

MQRMH - 参考消息头

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|---------------------------|----------------------------------|------------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>StrucLength</i> | MQRMH 的总长度，包括固定字段结尾的字符串，但不包括批量数据 | StrucLength |
| <i>Encoding</i> | 批量数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | 批量数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | 批量数据的格式名称 | FORMAT |
| <i>Flags</i> | 参考消息标志 | 标志 |
| <i>ObjectType</i> | 对象类型 | ObjectType |
| <i>ObjectInstanceId</i> | 对象实例标识 | ObjectInstanceId |
| <i>SrcEnvLength</i> | 源环境数据的长度 | SrcEnv 长度 |
| <i>SrcEnvOffset</i> | 源环境数据的偏移量 | SrcEnv 偏移量 |
| <i>SrcNameLength</i> | 源对象名的长度 | SrcName 长度 |
| <i>SrcNameOffset</i> | 源对象名的偏移量 | SrcName 偏移量 |
| <i>DestEnvLength</i> | 目标环境数据的长度 | DestEnv 长度 |
| <i>DestEnvOffset</i> | 目标环境数据的偏移量 | DestEnv 偏移量 |
| <i>DestNameLength</i> | 目标对象名的长度 | DestName 长度 |
| <i>DestNameOffset</i> | 目标对象名的偏移量 | DestName 偏移量 |
| <i>DataLogicalLength</i> | 批量数据的长度 | DataLogicalLength |
| <i>DataLogicalOffset</i> | 批量数据的低偏移 | DataLogicalOffset |
| <i>DataLogicalOffset2</i> | 批量数据的高偏移 | DataLogicalOffset2 |

MQRMH 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

用途: MQRMH 结构定义参考消息头的格式。此头与用户编写的消息通道出口配合使用，以将大量数据 (称为批量数据) 从一个队列管理器发送到另一个队列管理器。与正常消息传递不同的是，批量数据不会存储在队列上；相反，只有对批量数据的引用才会存储在队列上。这将降低 MQ 资源被少量超大消息耗尽的可能性。

格式名: MQFMT_REF_MSG_HEADER。

字符集和编码:MQRMH 中的字符数据以及由偏移量字段寻址的字符串必须位于本地队列管理器的字符集中; 这是由 *CodedCharSetId* queue-manager 属性提供的。MQRMH 中的数字数据必须采用本机编码; 这由 C 编程语言的 MQENC_NATIVE 值提供。

在以下位置将 MQRMH 的字符集和编码设置为 *CodedCharSetId* 和 *Encoding* 字段:

- MQMD (如果 MQRMH 结构位于消息数据的开头), 或者
- MQRMH 结构之前的头结构 (所有其他情况)。

用法: 应用程序放置由 MQRMH 组成的消息, 但忽略批量数据。当消息通道代理 (MCA) 从传输队列读取消息时, 将调用用户提供的消息出口来处理参考消息头。在 MCA 将消息通过通道发送到下一个队列管理器之前, 出口可以将 MQRMH 结构标识的批量数据附加到参考消息。

在接收端, 必须存在等待参考消息的消息出口。接收到参考消息时, 出口必须根据消息中 MQRMH 之后的批量数据创建对象, 然后在没有批量数据的情况下传递参考消息。稍后可由从队列中读取参考消息 (没有批量数据) 的应用程序检索参考消息。

通常, MQRMH 结构是消息中的所有结构。但是, 如果消息位于传输队列上, 那么 MQRMH 结构之前还有一个或多个头。

还可以将参考消息发送到分发列表。在这种情况下, 当消息位于传输队列中时, MQDH 结构及其相关记录在 MQRMH 结构之前。

注: 请勿将参考消息作为分段消息发送, 因为消息出口无法正确处理该消息。

数据转换: 出于数据转换目的, 转换 MQRMH 结构包括转换源环境数据, 源对象名称, 目标环境数据和目标对象名称。在结构开始的 *StrucLength* 字节内的任何其他字节都将被废弃或在数据转换后具有未定义的值。如果满足以下所有条件, 那么将转换批量数据:

- 执行数据转换时, 消息中存在批量数据。
- MQRMH 中的 *Format* 字段的值不是 MQFMT_NONE。
- 存在具有指定格式名称的用户编写的数据转换出口。

但是, 请注意, 当消息位于队列中时, 通常批量数据不 存在于消息中, 因此批量数据由 MQGMO_CONVERT 选项进行转换。

MQRMH 的字段

MQRMH 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

CodedCharSetId (MQLONG)

这指定批量数据的字符集标识; 它不适用于 MQRMH 结构本身中的字符数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。可以使用以下特殊值:

MQCCSI_INHERIT

遵循 此结构的数据中的字符数据与此结构位于同一字符集中。

队列管理器将消息中发送的结构中的此值更改为结构的实际字符集标识。如果未发生错误, 那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER, 请不要使用 MQCCSI_INHERIT。

此值在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

此字段的初始值为 MQCCSI_UNDEFINED。

DataLogical 长度 (MQLONG)

DataLogicalLength 字段指定 MQRMH 结构引用的批量数据的长度。

如果消息中实际存在批量数据, 那么数据将从 MQRMH 结构开头的 *StrucLength* 字节偏移量开始。整条消息的长度减去 *StrucLength* 给出了存在的批量数据的长度。

如果消息中存在数据，那么 *DataLogicalLength* 指定相关数据量。正常情况是 *DataLogicalLength* 具有与消息中存在的消息长度相同的值。

如果 MQRMH 结构表示对象中的剩余数据 (从指定的逻辑偏移量开始)，那么可以将值零用于 *DataLogicalLength*，前提是消息中实际上不存在批量数据。

如果不存在任何数据，那么 MQRMH 的结束与消息的结束一致。

此字段的初始值为 0。

DataLogicalOffset (MQLONG)

此字段指定从成批数据构成部分的对象开始的成批数据的低偏移量。从对象开始的批量数据的偏移称为逻辑偏移量。这不是从 MQRMH 结构开始的批量数据的物理偏移量；该偏移量由 *StrucLength* 提供。

为了允许使用参考消息发送大对象，逻辑偏移分为两个字段，实际逻辑偏移由这两个字段的总和给出：

- *DataLogicalOffset* 表示当逻辑偏移量除以 1 000 000 000 时获得的余数。因此，它是 0 到 999 999 999 范围内的值。
- *DataLogicalOffset2* 表示逻辑偏移量除以 1 000 000 000 时获得的结果。因此，它是逻辑偏移量中存在的 1 000 000 000 的完整倍数的数目。倍数数在范围 0 到 999 999 999 之间。

此字段的初始值为 0。

DataLogicalOffset2 (MQLONG)

此字段指定从成批数据构成部分的对象开始的成批数据的高偏移量。它是 0 到 999 999 999 范围内的值。请参阅 *DataLogicalOffset* 以了解详细信息。

此字段的初始值为 0。

DestEnv 长度 (MQLONG)

这是目标环境数据的长度。如果此字段为零，那么没有目标环境数据，将忽略 *DestEnvOffset*。

DestEnv 偏移量 (MQLONG)

此字段指定目标环境数据从 MQRMH 结构开始的偏移量。如果创建者知道目标环境数据，那么可以由参考消息的创建者指定该数据。例如，在 Windows 上，目标环境数据可能是要存储批量数据的对象的目录路径。但是，如果创建者不知道目标环境数据，那么由用户提供的消息出口负责确定所需的任何环境信息。

目标环境数据的长度由 *DestEnvLength* 给出；如果此长度为零，那么没有目标环境数据，并且将忽略 *DestEnvOffset*。如果存在，那么目标环境数据必须完全驻留在结构开始后的 *StrucLength* 字节内。

应用程序不得假定目标环境数据与 *SrcEnvOffset*，*SrcNameOffset* 和 *DestNameOffset* 字段所寻址的任何数据相邻。

此字段的初始值为 0。

DestName 长度 (MQLONG)

目标对象名的长度。如果此字段为零，那么没有目标对象名，将忽略 *DestNameOffset*。

DestName 偏移量 (MQLONG)

此字段指定目标对象名从 MQRMH 结构开始的偏移量。目标对象名可以由参考消息的创建者指定 (如果创建者知道该数据)。但是，如果创建者不知道目标对象名，那么用户提供的消息出口负责标识要创建或修改的对象。

目标对象名的长度由 *DestNameLength* 给出；如果此长度为零，那么没有目标对象名，并且将忽略 *DestNameOffset*。如果存在，那么目标对象名必须完全位于结构开头的 *StrucLength* 字节内。

应用程序不得假定目标对象名与 *SrcEnvOffset*，*SrcNameOffset* 和 *DestEnvOffset* 字段所寻址的任何数据都是连续的。

此字段的初始值为 0。

Encoding (MQLONG)

这指定批量数据的数字编码; 它不适用于 MQRMH 结构本身中的数字数据。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。

此字段的初始值为 MQENC_NATIVE。

Flags (MQLONG)

这些是参考消息标志。定义了以下标志:

MQRMHF_LAST

此标志指示参考消息表示或包含被引用对象的最后部分。

MQRMHF_NOT_LAST

参考消息不包含或表示对象的最后部分。MQRMHF_NOT_LAST 帮助程序文档。不打算将此选项与任何其他选项一起使用, 但由于其值为零, 因此无法检测到此类使用。

此字段的初始值为 MQRMHF_NOT_LAST。

Format (MQCHAR8)

这指定批量数据的格式名。

在 MQPUT 或 MQPUT1 调用上, 应用程序必须将此字段设置为适合于数据的值。此字段的编码规则与 MQMD 中 *Format* 字段的编码规则相同。

此字段的初始值为 MQFMT_NONE。

ObjectInstance 标识 (MQBYTE24)

使用此字段来标识对象的特定实例。如果不需要, 请将其设置为以下值:

MQOII_NONE

未指定对象实例标识。对于字段的长度, 该值为二进制零。

对于 C 编程语言, 还定义了常量 MQOII_NONE_ARRAY; 此值与 MQOII_NONE 相同, 但是字符数组而不是字符串。

此字段的长度由 MQ_OBJECT_INSTANCE_ID_LENGTH 提供。此字段的初始值为 MQOII_NONE。

ObjectType (MQCHAR8)

这是消息出口可用于识别其支持的参考消息类型的名称。该名称必须符合与上述 *Format* 字段相同的规则。

此字段的初始值为 8 空白。

SrcEnv 长度 (MQLONG)

源环境数据的长度。如果此字段为零, 那么没有源环境数据, 将忽略 *SrcEnvOffset*。

此字段的初始值为 0。

SrcEnv 偏移量 (MQLONG)

此字段指定源环境数据从 MQRMH 结构开始的偏移量。源环境数据可以由参考消息的创建者指定 (如果创建者知道该数据)。例如, 在 Windows 上, 源环境数据可能是包含批量数据的对象的目录路径。但是, 如果创建者不知道源环境数据, 那么用户提供的消息出口必须确定所需的任何环境信息。

源环境数据的长度由 *SrcEnvLength* 给出; 如果此长度为零, 那么没有源环境数据, 并且将忽略 *SrcEnvOffset*。如果存在, 那么源环境数据必须完全位于结构开始后的 *StrucLength* 字节内。

应用程序不得假定环境数据在结构中的最后一个固定字段之后立即启动, 或者它与 *SrcNameOffset*, *DestEnvOffset* 和 *DestNameOffset* 字段所寻址的任何数据相邻。

此字段的初始值为 0。

SrcName 长度 (MQLONG)

源对象名称的长度。如果此字段为零, 那么没有源对象名, 将忽略 *SrcNameOffset*。

此字段的初始值为 0。

SrcName 偏移量 (MQLONG)

此字段指定源对象名从 MQRMH 结构开始的偏移量。源对象名可由参考消息的创建者指定 (如果创建者知道该数据)。但是, 如果创建者不知道源对象名称, 那么用户提供的消息出口必须标识要访问的对象。

源对象名的长度由 SrcNameLength 给出; 如果此长度为零, 那么没有源对象名, 并且将忽略 SrcNameOffset。如果存在, 那么源对象名必须完全位于结构开始后的 StrucLength 字节内。

应用程序不得假定源对象名与 SrcEnvOffset, DestEnvOffset 和 DestNameOffset 字段所寻址的任何数据都是连续的。

此字段的初始值为 0。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQRMH_STRUC_ID

参考消息头结构的标识。

对于 C 编程语言, 还定义了常量 MQRMH_STRUC_ID_ARRAY; 此值与 MQRMH_STRUC_ID 相同, 但是字符数组而不是字符串。

此字段的初始值为 MQRMH_STRUC_ID。

StrucLength (MQLONG)

MQRMH 的总长度, 包括固定字段末尾的字符串, 但不包括批量数据。

此字段的初始值为零。

Version (MQLONG)

结构版本号。该值必须为:

MQRMH_VERSION_1

Version-1 参考消息头结构。

以下常量指定当前版本的版本号:

MQRMH_CURRENT_VERSION

参考消息头结构的当前版本。

此字段的初始值为 MQRMH_VERSION_1。

MQRMH 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|------------------|------------------|--------|
| StrucId | MQRMH_STRUC_ID | 'RMH~' |
| Version | MQRMH_VERSION_1 | 1 |
| StrucLength | None | 0 |
| Encoding | MQENC_NATIVE | 取决于环境 |
| CodedCharSetId | MQCCSI_UNDEFINED | 0 |
| Format | MQFMT_NONE | 空白 |
| Flags | MQRMHF_NOT_LAST | 0 |
| ObjectType | None | 空白 |
| ObjectInstanceId | MQOII_NONE | Null |
| SrcEnvLength | None | 0 |

表 542: MQRMH 的 MQRMH 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|---------------------------|-------|-----|
| <i>SrcEnvOffset</i> | None | 0 |
| <i>SrcNameLength</i> | None | 0 |
| <i>SrcNameOffset</i> | None | 0 |
| <i>DestEnvLength</i> | None | 0 |
| <i>DestEnvOffset</i> | None | 0 |
| <i>DestNameLength</i> | None | 0 |
| <i>DestNameOffset</i> | None | 0 |
| <i>DataLogicalLength</i> | None | 0 |
| <i>DataLogicalOffset</i> | None | 0 |
| <i>DataLogicalOffset2</i> | None | 0 |

注意:

1. 符号 `~` 表示单个空白字符。
2. 在 C 编程语言中, 宏变量 `MQRMH_DEFAULT` 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQRMH MyRMH = {MQRMH_DEFAULT};
```

C 声明

```
typedef struct tagMQRMH MQRMH;
struct tagMQRMH {
    MQCHAR4   StructId;           /* 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 声明

```
** 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-DATALOGICALENGTH 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.

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

MQRMH                DSECT
MQRMH_STRUCID        DS    CL4  Structure identifier
MQRMH_VERSION        DS    F    Structure version number
MQRMH_STRUCLNGTH     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_DATALOGICALENGTH 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)

```

Visual Basic 声明

```

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-响应记录

下表汇总了结构中的字段。

| 表 543: MQRR 中的字段 | | |
|------------------|---------|--------------------------|
| 字段 | 描述 | Topic |
| <i>CompCode</i> | 队列的完成代码 | CompCode |
| <i>Reason</i> | 队列的原因码 | 原因 |

MQRR 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

用途: 当目标是分发列表时, 使用 MQRR 结构接收由单个目标队列的打开或放置操作生成的完成代码和原因码。MQRR 是 MQOPEN, MQPUT 和 MQPUT1 调用的输出结构。

字符集和编码: MQRR 中的数据必须在由 *CodedCharSetId* queue-manager 属性提供的字符集中, 并且必须在由 MQENC_NATIVE 提供的本地队列管理器的编码中。但是, 如果应用程序作为 MQ MQI 客户机运行, 那么该结构必须采用客户机的字符集和编码。

用法: 通过在 MQOPEN 和 MQPUT 调用上提供这些结构的数组, 或者在 MQPUT1 调用上提供这些结构的数组, 您可以在混合调用结果时确定分发列表中所有队列的完成代码和原因码, 即, 调用在列表中某些队列成功但其他队列失败时。来自调用的原因码 MQRC_MULTIPLE_REASON 指示响应记录 (如果由应用程序提供) 已由队列管理器设置。

MQRR 的字段

MQRR 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

CompCode (MQLONG)

这是队列的打开或放置操作生成的完成代码, 其名称由 MQOPEN 或 MQPUT1 调用上提供的 MQOR 结构数组中的相应元素指定。

这始终是输出字段。此字段的初始值为 MQCC_OK。

原因 (MQLONG)

这是队列的打开或放入操作产生的原因码, 其名称由 MQOPEN 或 MQPUT1 调用上提供的 MQOR 结构数组中的相应元素指定。

这始终是输出字段。此字段的初始值为 MQRC_NONE。

MQRR 的初始值和语言声明

| 表 544: MQRR 的 MQRR 中字段的初始值 | | |
|----------------------------|-----------|-----|
| 字段名称 | 常量的名称 | 常量值 |
| CompCode | MQCC_OK | 0 |
| Reason | MQRC_NONE | 0 |

注意:

- 在 C 编程语言中, 宏变量 MQRR_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQRR MyRR = {MQRR_DEFAULT};
```

C 声明

```
typedef struct tagMQRR MQRR;  
struct tagMQRR {  
    MQLONG CompCode; /* Completion code for queue */  
    MQLONG Reason; /* Reason code for queue */  
};
```

COBOL 声明

```
** 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.

```

PL/I 声明

```

dcl
  1 MQRR based,
  3 CompCode fixed bin(31), /* Completion code for queue */
  3 Reason   fixed bin(31); /* Reason code for queue */

```

Visual Basic 声明

```

Type MQRR
  CompCode As Long 'Completion code for queue'
  Reason   As Long 'Reason code for queue'
End Type

```

MQSCO-SSL 配置选项

下表汇总了结构中的字段。

| 表 545: 字段(F) MQSCO. | | |
|---|---------------------------------|---|
| MQSCO 中的字段列表 (按版本), 以及指向描述这些字段的主题链接。 | | |
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>KeyRepository</i> | 密钥存储库的位置 | KeyRepository |
| <i>CryptoHardware</i> | 加密硬件的详细信息 | CryptoHardware |
| <i>AuthInfoRecCount</i> | 存在的 MQAIR 记录数 | AuthInfoRecCount |
| <i>AuthInfoRecOffset</i> | 第一个 MQAIR 记录与 MQSCO 的开始处之间的偏移量 | AuthInfoRecOffset |
| <i>AuthInfoRecPtr</i> | 第一个 MQAIR 记录的地址 | AuthInfoRecPtr |
| 注: 如果 <i>Version</i> 小于 MQSCO_VERSION_2, 那么将忽略以下两个字段。 | | |
| <i>KeyResetCount</i> | SSL 密钥重置计数 | KeyReset 计数 |
| <i>FipsRequired</i> | 在 WebSphere MQ 中使用 FIPS 认证的密码算法 | 第 487 页的『FipsRequired (MQLONG)』 |
| 注: 如果 <i>Version</i> 小于 MQSCO_VERSION_3, 那么将忽略以下字段。 | | |
| <i>EncryptionPolicySuiteB</i> | 仅使用 Suite B 密码算法 | EncryptionPolicySuiteB |
| 注: 如果 <i>Version</i> 小于 MQSCO_VERSION_4, 那么将忽略以下字段。 | | |
| <i>CertificateValPolicy</i> | 证书确认策略 | CertificateVal 策略 |

相关参考

第 283 页的『MQCNO - 连接选项』

下表汇总了结构中的字段。

[第 486 页的『MQSCO 概述』](#)

可用性: AIX, HP-UX, IBM i, Solaris, Linux 和 Windows 客户机。

[第 486 页的『MQSCO 的字段』](#)

[第 489 页的『MQSCO 的初始值和语言声明』](#)

MQSCO 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux 和 Windows 客户机。

用途:MQSCO 结构 (与 MQCD 结构中的 SSL 字段结合使用) 允许作为 WebSphere MQ MQI 客户机运行的应用程序指定配置选项, 以在通道协议为 TCP/IP 时控制对客户机连接使用 SSL。该结构是 MQCONN 调用上的输入参数。

如果客户机通道的通道协议不是 TCP/IP, 那么将忽略 MQSCO 结构。

字符集和编码:MQSCO mustis 中由 *CodedCharSetId* queue-manager 属性提供的字符集中的数据以及由 MQENC_NATIVE 提供的本地队列管理器的编码。

MQSCO 的字段

MQSCO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

AuthInfoRecCount (MQLONG)

这是由 *AuthInfoRecPtr* 或 *AuthInfoRecOffset* 字段寻址的认证信息 (MQAIR) 记录数。有关更多信息, 请参阅第 242 页的『MQAIR-认证信息记录』。值必须大于等于零。如果该值无效, 那么调用将失败, 原因码为 MQRC_AUTH_INFO_REC_COUNT_ERROR。

这是一个输入字段。此字段的初始值为 0。

AuthInfoRecOffset (MQLONG)

这是从 MQSCO 结构开始的第一个认证信息记录的偏移量 (以字节为单位)。偏移可以是正数或负数。如果 *AuthInfoRecCount* 为零, 那么将忽略该字段。

您可以使用 *AuthInfoRecOffset* 或 *AuthInfoRecPtr* 来指定 MQAIR 记录, 但不能同时指定这两者; 请参阅 *AuthInfoRecPtr* 字段的描述以获取详细信息。

这是一个输入字段。此字段的初始值为 0。

AuthInfoRecPtr (PMQAIR)

这是第一个认证信息记录的地址。如果 *AuthInfoRecCount* 为零, 那么将忽略该字段。

您可以通过以下两种方式之一提供 MQAIR 记录的数组:

- 通过使用指针字段 *AuthInfoRecPtr*

在这种情况下, 应用程序可以声明与 MQSCO 结构不同的 MQAIR 记录数组, 并将 *AuthInfoRecPtr* 设置为该数组的地址。

请考虑将 *AuthInfoRecPtr* 用于以可移植到不同环境 (例如 C 编程语言) 的方式支持指针数据类型的编程语言。

- 通过使用偏移量字段 *AuthInfoRecOffset*

在这种情况下, 应用程序必须声明包含后跟 MQAIR 记录数组的 MQSCO 的复合结构, 并将 *AuthInfoRecOffset* 设置为数组中第一个记录从 MQSCO 结构开始的偏移量。确保此值正确, 并且具有可在 MQLONG 中容纳的值 (最严格的编程语言是 COBOL, 其有效范围为 -999 999 999 到 +999 999 999 999)。

请考虑将 *AuthInfoRecOffset* 用于不支持指针数据类型的编程语言, 或以无法移植到不同环境 (例如 COBOL 编程语言) 的方式实现指针数据类型的编程语言。

无论您选择何种技术，都只能使用 *AuthInfoRecPtr* 和 *AuthInfoRecOffset* 之一；如果两者都非零，那么调用将失败，原因码为 MQRC_AUTH_INFO_REC_ERROR。

这是一个输入字段。此字段的初始值是那些支持指针的编程语言中的空指针，否则为全空字节字符串。

注：在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串。

CertificateVal 策略 (MQLONG)

此字段指定使用的证书验证策略的类型。可以将该字段设置为下列其中一个值：

MQ_CERT_VAL_POLICY_ANY

应用安全套接字库支持的每个证书验证策略。如果任何策略认为证书链有效，请接受该证书链。

MQ_CERT_VAL_POLICY_RFC5280

仅应用符合 RFC5280 的证书验证策略。此设置提供比 ANY 设置更严格的验证，但是会拒绝一些较旧的数字证书。

此字段的初始值为 MQ_CERT_VAL_POLICY_ANY

CryptoHardware (MQCHAR256)

此字段提供连接到客户机系统的加密硬件的配置详细信息。

将该字段设置为以下格式的字符串，或者将其留空或为空：

```
GSK_PKCS11=<the PKCS #11 driver path and file name>;<the PKCS #11 token label>;<the PKCS #11 token password>;<symmetric cipher setting>;
```

要使用符合 PKCS #11 接口的加密硬件，例如 IBM 4960 或 IBM 4764，必须指定 PKCS #11 驱动程序路径，PKCS #11 令牌标签和 PKCS #11 令牌密码字符串，每个都以分号终止。

PKCS #11 驱动程序路径是提供 PKCS #11 卡支持的共享库的绝对路径。PKCS #11 驱动程序文件名是共享库的名称。PKCS #11 路径和文件名所需的值的示例为：

```
/usr/lib/pkcs11/PKCS11_API.so
```

PKCS #11 令牌标签必须完全为小写。如果您已使用混合大小写或大写令牌标签配置硬件，请使用此小写标签重新配置硬件。

如果不需要加密硬件配置，请将该字段设置为空白或空。

如果该值比字段的长度短，那么用空字符终止该值，或用空白填充该字段的长度。如果该值无效，或者在用于配置加密硬件时导致失败，那么调用将失败，原因码为 MQRC_CRYPTO_HARDWARE_ERROR。

这是一个输入字段。此字段的长度由 MQ_SSL_CRYPTO_HARDWARE_LENGTH 提供。此字段的初始值是 C 中的空字符串，以及其他编程语言中的空白字符。

EncryptionPolicySuiteB(MQLONG)

此字段指定是否使用符合 Suite B 的密码术以及采用的强度级别。该值可以是下列其中一项或多项：

- MQ_SUITE_B_NONE
不使用符合套件 B 的密码术。
- MQ_SUITE_B_128_BIT
使用套件 B 128 位强度安全性。
- MQ_SUITE_B_192_BIT
使用套件 B 192 位强度安全性。

注：将 MQ_SUITE_B_NONE 与此字段中的任何其他值配合使用无效。

FipsRequired (MQLONG)

可以使用加密硬件配置 WebSphere MQ，以便所使用的加密模块是由硬件产品提供的模块；这些模块可以通过 FIPS 认证到特定级别，具体取决于所使用的加密硬件产品。使用此字段来指定如果在 WebSphere MQ 提供的软件中提供了密码术，那么仅使用 FIPS 认证的算法。

安装 WebSphere MQ 时，还会安装 SSL 密码术的实现，这将提供一些经 FIPS 认证的模块。

值可以是：

MQSSL_FIPS_NO

这是缺省值。设置为该值时：

- 可以使用特定平台上支持的任何 CipherSpec。
- 如果在不使用加密硬件的情况下运行，那么以下 CipherSpecs 在 WebSphere MQ 平台上使用 FIPS 140-2 认证的密码术运行：
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES

设置为该值时，除非您使用加密硬件来执行密码术，否则可以确保

- 此客户机连接所应用的 CipherSpec 中只能使用 FIPS 认证的密码算法。
- 仅当使用了下列其中一个密码规范时，进站和出站 SSL 通道连接才会成功：
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

注意：

1. 不推荐 CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA。
2. 在可能的情况下，如果配置了仅 FIPS CipherSpecs，那么 MQI 客户机将拒绝使用 MQRC_SSL_INITIALIZATION_ERROR 指定非 FIPS CipherSpec 的连接。WebSphere MQ 不保证拒绝所有此类连接，您负责确定 WebSphere MQ 配置是否符合 FIPS 标准。

distributed *KeyRepository (MQCHAR256)*

此字段仅适用于在 UNIX，Linux 和 Windows 系统上运行的 WebSphere MQ MQI 客户机。它指定用于存储密钥和证书的密钥数据库文件的位置。密钥数据库文件必须具有格式为 zzz.kdb 的文件名，其中 zzz 是用户可选择的。*KeyRepository* 字段包含此文件的路径以及文件名词干（文件名中的所有字符最多但不包括最后的 .kdb）。将自动添加 .kdb 文件后缀。

每个密钥数据库文件都有关联的密码隐藏文件。这将保存用于允许程序化访问密钥数据库的编码密码。密码隐藏文件必须位于同一目录中，并且具有与密钥数据库相同的文件系统，并且必须以后缀 .sth 结尾。

例如，如果 *KeyRepository* 字段具有值 /xxx/yyy/key，那么密钥数据库文件必须为 /xxx/yyy/key.kdb，密码存储文件必须为 /xxx/yyy/key.sth，其中 xxx 和 yyy 表示目录名称。

如果该值比字段的长度短，那么用空字符终止该值，或用空白填充该字段的长度。未检查该值；如果访问密钥存储库时发生错误，那么调用将失败，原因码为 MQRC_KEY_REPOSITORY_ERROR。

要从 WebSphere MQ MQI 客户机运行 SSL 连接，请将 *KeyRepository* 设置为有效的密钥数据库文件名。

这是一个输入字段。此字段的长度由 MQ_SSL_KEY_REPOSITORY_LENGTH 提供。此字段的初始值是 C 中的空字符串，以及其他编程语言中的空白字符。

KeyResetCount (MQLONG)

这表示在重新协商密钥之前在 SSL 或 TLS 对话中发送和接收的未加密字节总数。

此字节数包括由 MCA 发送的控制信息。

如果指定 1 字节到 32 KB 范围内的 SSL 或 TLS 密钥重置计数，那么 SSL 或 TLS 通道将使用 32 KB 的密钥重置计数。这是为了避免对于小型 SSL 或 TLS 密钥重置值将发生的过多密钥重置的处理成本。

这是一个输入字段。该值是范围在 0 到 999 999 999 999 之间的数字，缺省值为 0。使用值 0 指示永不重新协商密钥。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQSCO_STRUC_ID

SSL 配置选项结构的标识。

对于 C 编程语言, 还定义了常量 MQSCO_STRUC_ID_ARRAY; 这具有与 MQSCO_STRUC_ID 相同的值, 但它是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQSCO_STRUC_ID。

Version (MQLONG)

这是结构版本号; 值必须为:

MQSCO_VERSION_1

Version-1 SSL 配置选项结构。

MQSCO_VERSION_2

Version-2 SSL 配置选项结构。

MQSCO_VERSION_3

Version-3 SSL 配置选项结构。

MQSCO_VERSION_4

Version-4 SSL 配置选项结构。

以下常量指定当前版本的版本号:

MQSCO_CURRENT_VERSION

当前版本的 SSL 配置选项结构。

这始终是一个输入字段。此字段的初始值为 MQSCO_VERSION_1。

MQSCO 的初始值和语言声明

| 表 546: MQSCO 中字段的初始值. | | |
|-------------------------------|--|---------------------|
| MQSCO 中字段及其初始值的描述 | | |
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQSCO_STRUC_ID | 'SCO~' |
| <i>Version</i> | MQSCO_CURRENT_VERSION | 1 |
| <i>KeyRepository</i> | None | 空字符串或空白 |
| <i>CryptoHardware</i> | None | 空字符串或空白 |
| <i>AuthInfoRecCount</i> | None | 0 |
| <i>AuthInfoRecOffset</i> | None | 0 |
| <i>AuthInfoRecPtr</i> | None | 空指针或空字节 |
| <i>KeyResetCount</i> | MQSCO_RESET_COUNT_DEFAULT | 0 |
| <i>FipsRequired</i> | MQSSL_FIPS_NO | 0 |
| <i>EncryptionPolicySuiteB</i> | MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE | 1, 0, 0, 0 |
| <i>CertificateValPolicy</i> | 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};
```

C 声明

```
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 */
                                           /* algorithms */
/* Ver:2 */
    MQLONG     EncryptionPolicySuiteB[4]; /* Use only Suite B */
/* Ver:3 */
    MQLONG     CertificateValPolicy;  /* cryptographic algorithms */
                                           /* Certificate validation */
                                           /* policy */
/* Ver:4 */
}
```

COBOL 声明

```
** 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-AUTHINFORECPTN 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
```

PL/I 声明

```

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 */

```

Visual Basic 声明

```

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-预订描述符

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|----------------------------|--------|-------------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 选项 | 选项 |
| <i>ObjectName</i> | 对象名 | ObjectName |
| <i>AlternateUserId</i> | 备用用户标识 | AlternateUserId |
| <i>AlternateSecurityId</i> | 备用安全标识 | AlternateSecurityId |
| <i>SubExpiry</i> | 预订到期时间 | SubExpiry |
| <i>ObjectString</i> | 对象字符串 | ObjectString |
| <i>SubName</i> | 预订名称 | SubName |
| <i>SubUserData</i> | 预订用户数据 | SubUserData |
| <i>SubCorrelId</i> | 预订相关标识 | SubCorrelId |
| <i>PubPriority</i> | 发布优先级 | PubPriority |
| <i>PubAccountingToken</i> | 发布记帐令牌 | PubAccountingToken |

| 字段 | 描述 | Topic |
|---------------------------|------------|------------------------------------|
| <i>PubAppIdentityData</i> | 发布应用程序身份数据 | PubAppIdentityData |
| <i>SelectionString</i> | 提供选择条件的字符串 | SelectionString |
| <i>SubLevel</i> | 预订级别 | SubLevel |
| <i>ResObjectString</i> | 长对象名 | ResObject 字符串 |

MQSD 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途:MQSD 结构用于指定有关正在进行的预订的详细信息。

该结构是 MQSUB 调用上的输入/输出参数。有关更多信息, 请参阅 [MQSUB 使用说明](#)。

受管预订: 如果应用程序不特定需要使用特定队列作为与其预订匹配的发布的目标, 那么它可以使用受管预订功能。如果应用程序选择使用受管预订, 那么队列管理器会通过提供对象句柄作为 MQSUB 调用的输出, 向订户通知发送已发布消息的目标。有关更多信息, 请参阅 [Hobj \(MQHOBJ\)-输入/输出](#)。

除去预订时, 队列管理器还会承诺在以下情况下清除未从受管目标检索到的消息:

- 通过将 MQCLOSE 与 MQCO_REMOVE_SUB 配合使用并关闭受管 Hobj 来除去预订时。
- 通过隐式方式, 当使用非持久预订 (MQSO_NON_持久) 与应用程序断开连接时
- 在由于预订已到期且受管 Hobj 已关闭而将其除去时到期。

必须将受管预订与非持久预订配合使用, 以便可以执行此清除操作, 从而使已关闭的非持久预订的消息不会占用队列管理器中的空间。持久预订还可以使用受管目标。

版本:MQSD 的当前版本为 MQSD_VERSION_1。

字符集和编码:MQSD 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集, 以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是, 如果应用程序作为 MQ MQI 客户机运行, 那么该结构必须采用客户机的字符集和编码。

MQSD 的字段

MQSD 结构包含以下字段; 这些字段按字母顺序描述:

AlternateSecurity 标识 (MQBYTE40)

这是与 *AlternateUser* 标识一起传递给授权服务的安全标识, 以允许执行相应的授权检查。

仅当指定了 MQSO_ALTERNATE_USER_AUTHORITY 时, 才会使用 *AlternateSecurityId*, 并且 *AlternateUserId* 字段并非完全空白, 直到第一个空字符或字段结束。

从使用 MQSO_RESUME 的 MQSUB 调用返回时, 此字段保持不变。

请参阅 MQOD 数据类型中 [第 416 页的『AlternateSecurity 标识 \(MQBYTE40\)』](#) 的描述以获取更多信息。

AlternateUser 标识 (MQCHAR12)

如果指定 MQSO_ALTERNATE_USER_AUTHORITY, 那么此字段包含备用用户标识, 用于检查预订的授权以及目标队列的输出 (在 MQSUB 调用的 *Hobj* 参数中指定), 以代替应用程序当前正在运行的用户标识。

如果成功, 那么此字段中指定的用户标识将记录为拥有预订的用户标识, 以代替应用程序当前正在运行的用户标识。

如果指定了 MQSO_ALTERNATE_USER_AUTHORITY, 并且此字段完全为空白, 直到第一个空字符或字段结束, 那么仅当不需要用户授权即可使用指定的选项或输出的目标队列来预订此主题时, 预订才能成功。

如果未指定 MQSO_ALTERNATE_USER_AUTHORITY，那么将忽略此字段。

在所指示的环境中存在以下差异：

- 在 z/OS 上，仅使用 AlternateUser 标识的前 8 个字符来检查预订的授权。但是，必须授权当前用户标识指定此特定备用用户标识；备用用户标识的所有 12 个字符都用于此检查。用户标识必须仅包含外部安全管理器允许的字符。

从使用 MQSO_RESUME 的 MQSUB 调用返回时，此字段保持不变。

这是一个输入字段。此字段的长度由 MQ_USER_ID_LENGTH 给出。此字段的初始值是 C 中的空字符串，以及其他编程语言中的 12 个空白字符。

ObjectName (MQCHAR48)

这是在本地队列管理器上定义的主题对象的名称。

此名称可包含以下字符：

- 大写字母字符 (A 到 Z)
- 小写字母字符 (a 到 z)
- 数字位 (0 到 9)
- 句点 (.)、正斜杠 (/)、下划线 (_)、百分号 (%)

此名称不得包含前导空格或嵌入空格，但可以包含尾部空格。使用空字符来指示名称中重要数据的结束；空字符及其后的任何字符被视为空白。以下限制适用于指示的环境中：

- 在使用 EBCDIC 片假名的系统上，不得使用小写字符。
- 在 z/OS 上：
 - 避免使用以下划线开头或结尾的名称；操作和控制面板无法处理这些名称。
 - 百分号字符对于 RACF 具有特殊含义。如果使用 RACF 作为外部安全性管理器，那么名称不得包含百分比。如果这样做，那么在使用 RACF 通用概要文件时，这些名称不会包含在任何安全性检查中。
- 在 IBM i 上，当在命令上指定时，必须将包含小写字符，正斜杠或百分号的名称括在引号中。对于在结构中作为字段出现的名称或在调用时作为参数出现的名称，不得指定这些引号。

ObjectName 用于构成完整主题名称。

可以从两个不同的字段构建完整主题名称：*ObjectName* 和 *ObjectString*。有关如何使用这两个字段的详细信息，请参阅第 504 页的『使用主题字符串』。

如果找不到 *ObjectName* 字段所标识的对象，那么调用将失败，原因码为 MQRC_UNKNOWN_OBJECT_NAME，即使在 *ObjectString* 中指定了字符串也是如此。

从使用 MQSO_RESUME 选项的 MQSUB 调用返回时，此字段保持不变。

此字段的长度由 MQ_TOPIC_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

如果使用 MQSO_ALTER 选项改变现有预订，那么无法更改所预订的主题对象的名称。可以省略此字段和 *ObjectString* 字段。如果提供了它们，那么它们必须解析为相同的完整主题名称。如果它们不存在，那么调用将失败，并返回 MQRC_TOPIC_NOT_ALTERABLE。

ObjectString (MQCHARV)

这是要使用的长对象名。

ObjectString 用于构成完整主题名称。

可以从两个不同的字段构建完整主题名称：*ObjectName* 和 *ObjectString*。有关如何使用这两个字段的详细信息，请参阅第 504 页的『使用主题字符串』。

ObjectString 的最大长度为 10240。

如果未正确指定 *ObjectString*，那么根据如何使用 MQCHARV 结构的描述，或者如果该结构超过最大长度，那么调用将失败，原因码为 MQRC_OBJECT_STRING_ERROR。

这是一个输入字段。此结构中字段的初始值与 MQCHARV 结构中的初始值相同。

如果 *ObjectString* 中存在通配符，那么可以使用 MQSD 的 "选项" 字段中指定的通配符选项来控制这些通配符的解释。

从使用 MQSO_RESUME 选项的 MQSUB 调用返回时，此字段保持不变。如果提供了缓冲区，那么将在 *ResObjectString* 字段中返回所使用的完整主题名称。

如果使用 MQSO_ALTER 选项更改现有预订，那么无法更改预订的主题对象的长名称。可以省略此字段和 *ObjectName* 字段。如果提供了它们，那么它们必须解析为相同的完整主题名称，否则调用将失败并返回 MQRC_TOPIC_NOT_ALTERABLE。

选项 (MQLONG)

这提供了用于控制 MQSUB 调用的操作的选项。

必须至少指定下列其中一个选项:

- MQSO_ALTER
- MQSO_RESUME
- MQSO_CREATE

可以通过以下方式使用您为选项指定的值:

- 可以添加值。请勿多次添加同一常量。
- 如果编程语言支持按位运算，那么可以使用按位 OR 运算来组合值。

本主题中记录了无效的组合; 任何其他组合都是有效的。

访问权或创建选项: 访问权和创建选项控制是创建预订，还是返回或变更现有预订。必须至少指定其中一个选项。此表显示访问权和创建选项的有效组合。

| 选项组合 | 注意 |
|---|--|
| MQSO_CREATE | 如果不存在预订，那么创建预订。如果预订存在，那么此组合将失败。 |
| MQSO_RESUME | 恢复现有预订。如果不存在预订，那么此组合将失败。 |
| MQSO_CREATE + MQSO_RESUME | 如果预订不存在，那么创建预订，如果存在，那么恢复匹配的预订。当在运行多次的应用程序中使用此组合时，此组合很有用。 |
| MQSO_ALTER (请参阅注释) | 恢复现有预订，改变任何字段以与 MQSD 中指定的字段匹配。如果不存在预订，那么此组合将失败。 |
| MQSO_CREATE + MQSO_ALTER (请参阅注释) | 如果不存在预订，那么创建预订，如果存在预订，那么恢复匹配的预订，从而改变任何字段以与 MQSD 中指定的字段相匹配。在要确保其预订处于特定状态的应用程序中使用此组合时，此组合是有用的组合。 |
| <p>注: 指定 MQSO_ALTER 的选项还可以指定 MQSO_RESUME，但此组合对于单独指定 MQSO_ALTER 没有其他影响。MQSO_ALTER 意味着 MQSO_RESUME，因为调用 MQSUB 以变更预订意味着该预订也将恢复。但事实并非如此: 恢复预订并不意味着要进行更改。</p> | |

MQSO_CREATE

为指定的主题创建新预订。如果存在使用相同 *SubName* 的预订，那么调用将失败并返回 MQRC_SUB_ALREADY_EXISTS。可通过将 MQSO_CREATE 选项与 MQSO_RESUME 组合使用来避免此故障。并非总是需要 *SubName*。有关更多详细信息，请参阅该字段的描述。

将 MQSO_CREATE 与 MQSO_RESUME 组合后，如果找到指定 *SubName* 的预先存在的预订，那么将返回一个句柄；如果没有现有预订，那么将使用 MQSD 中提供的所有字段创建新的预订。

MQSO_CREATE 还可以与 MQSO_ALTER 结合使用，以达到类似的效果。

MQSO_RESUME

将句柄返回到与 *SubName* 指定的预订匹配的预先存在的预订。不会对匹配的预订属性进行任何更改，并且会在 MQSD 结构中的输出上返回这些属性。仅使用以下 MQSD 字段: StrucId, Version, Options, AlternateUserId 和 AlternateSecurityId 以及 SubName。

如果不存在与完整预订名称匹配的预订，那么调用将失败，原因码为 MQRC_NO_SUBSCRIPTION。可通过将 MQSO_CREATE 选项与 MQSO_RESUME 组合使用来避免此故障。

预订的用户标识是创建该预订的用户标识，或者如果该预订后来被其他用户标识变更，那么它是最近成功变更的用户标识。如果使用了 AlternateUser 标识，并且允许该用户使用备用用户标识，那么备用用户标识将记录为创建预订的用户标识，而不是用于创建预订的用户标识。

如果存在在没有 MQSO_ANY_USERID 选项的情况下创建的匹配预订，并且预订的用户标识与请求预订句柄的应用程序的用户标识不同，那么调用将失败，原因码为 MQRC_IDENTITY_MATCH。

如果存在匹配的预订并且当前正在使用该预订，那么调用将失败并返回 MQRC_SUBSCRIPTION_IN_USE。

如果 SubName 中指定的预订不是从应用程序恢复或变更的有效预订，那么调用将失败并返回 MQRC_INVALID_SUBSCRIPTION。

MQSO_ALTER 暗示了 MQSO_RESUME，因此您不需要将其与该选项结合使用。但是，组合这两个选项不会导致错误。

MQSO_ALTER

将句柄返回到与 *SubName* 中的名称指定的完整预订名称匹配的预先存在的预订。预订中与 MQSD 中指定的属性不同的任何属性都将在预订中更改，除非不允许对该属性进行更改。详细信息在每个属性的描述中进行了说明，并在下表中进行了汇总。如果尝试更改无法更改的属性，或更改已设置 MQSO_IMMUTABLE 选项的预订，那么调用将失败，原因码如下表中所示。

如果不存在与完整预订名称匹配的预订，那么调用将失败，原因码为 MQRC_NO_SUBSCRIPTION。可以通过将 MQSO_CREATE 选项与 MQSO_ALTER 组合使用来避免此故障。

将 MQSO_CREATE 与 MQSO_ALTER 组合会将句柄返回到指定 *SubName* 的预先存在的预订 (如果找到)；如果没有现有预订，那么将使用 MQSD 中提供的所有字段创建新的预订。

预订的用户标识是创建该预订的用户标识，或者如果该用户标识后来被其他用户标识更改，那么该用户标识是最新的成功更改的用户标识。如果使用了 AlternateUser 标识，并且允许该用户使用备用用户标识，那么会将备用用户标识记录为创建预订的用户标识，而不是用于创建预订的用户标识。

如果存在在没有选项 MQSO_ANY_USERID 的情况下创建的匹配预订，并且预订的用户标识与请求预订句柄的应用程序的用户标识不同，那么调用将失败，原因码为 MQRC_IDENTITY_MATCH。

如果存在匹配的预订并且当前正在使用该预订，那么调用将失败并返回 MQRC_SUBSCRIPTION_IN_USE。

如果 SubName 中指定的预订不是从应用程序恢复或变更的有效预订，那么调用将失败并返回 MQRC_INVALID_SUBSCRIPTION。

下表显示 MQSO_ALTER 更改 MQSD 和 MQSUB 中的属性值的能力。

| 数据类型描述符或函数调用 | 字段名称 | 能否使用 MQSO_ALTER 更改此属性 | 原因码 |
|--------------|-------|-----------------------|---|
| MQSD | 耐久性选项 | 否 | MQRC_DURABILITY_NOT_ALTERABLE |
| MQSD | 目标选项 | Yes | None |
| MQSD | 注册选项 | 是 (请参阅注释 第 496 页的『1』) | MQRC_GROUPING_NOT_ALTERABLE (如果尝试更改 MQSO_GROUP_SUB) |

| 表 547: MQSD 和 MQSUB 中可更改的属性 (继续) | | | |
|----------------------------------|---------------------|-----------------------|-------------------------------------|
| 数据类型描述符或函数调用 | 字段名称 | 能否使用 MQSO ALTER 更改此属性 | 原因码 |
| MQSD | 发布选项 | 是 (请参阅注释 第 496 页的『2』) | None |
| MQSD | 通配符选项 | 否 | MQRC_TOPIC_NOT_ALTERABLE |
| MQSD | 其他选项 | 否 (请参阅注释 第 496 页的『3』) | None |
| MQSD | ObjectName | 否 | MQRC_TOPIC_NOT_ALTERABLE |
| MQSD | AlternateUserId | 否 (请参阅注释 第 496 页的『4』) | None |
| MQSD | AlternateSecurityId | 否 (请参阅注释 第 496 页的『4』) | None |
| MQSD | SubExpiry | Yes | None |
| MQSD | ObjectString | 否 | MQRC_TOPIC_NOT_ALTERABLE |
| MQSD | SubName | 否 (请参阅注释 第 496 页的『5』) | None |
| MQSD | SubUserData | Yes | None |
| MQSD | SubCorreId | 是 (请参阅注释 第 496 页的『6』) | 在分组预订中时 MQRC_GROUPING_NOT_ALTERABLE |
| MQSD | PubPriority | Yes | None |
| MQSD | PubAccountingToken | Yes | None |
| MQSD | PubAppIdentityData | Yes | None |
| MQSD | SubLevel | 否 | MQRC_SUBLEVEL_NOT_ALTERABLE |
| MQSUB | Hobj | 是 (请参阅注释 第 496 页的『6』) | 在分组预订中时 MQRC_GROUPING_NOT_ALTERABLE |

注意:

- 无法变更 MQSO_GROUP_SUB。
- 无法变更 MQSO_NEW_publicATIONS_ONLY, 因为它不是预订的一部分
- 这些选项不是预订的一部分
- 此属性不是预订的一部分
- 此属性是要变更的预订的身份
- 可更改, 但当部分分组子 (MQSO_GROUP_SUB) 时除外

耐久性选项: 以下选项控制预订的持久程度。只能指定其中一个选项。如果要使用 MQSO ALTER 选项更改现有预订, 那么无法更改预订的耐久性。从使用 MQSO_RESUME 的 MQSUB 调用返回时, 将设置相应的耐久性选项。

MQSO_持久

请求保留此主题的预订, 直到使用带有 MQCO_REMOVE_SUB 选项的 MQCLOSE 将其显式除去为止。如果未显式除去此预订, 那么即使在此应用程序与队列管理器的连接关闭后, 此预订也将保留。

如果对定义为不允许持久预订的主题请求持久预订, 那么调用将失败并返回 MQRC_DURABILITY_NOT_ALLOWED。

MQSO_non_持久

请求在关闭与队列管理器的应用程序连接时除去此主题的预订 (如果尚未显式除去)。MQSO_NON_耐久性与 MQSO_耐久性选项相反, 并且定义为帮助程序文档。如果两者都未指定, 那么它是缺省值。

目标选项: 以下选项控制将已预订的主题的发布发送到的目标。如果使用 MQSO ALTER 选项改变现有预订, 那么可以更改用于预订发布的目标。从使用 MQSO_RESUME 的 MQSUB 调用返回时, 将设置此选项 (如果适用)。

MQSO_MANAGED

请求将发布发送到的目标由队列管理器管理。

Hobj 中返回的对象句柄表示队列管理器受管队列，用于后续 MQGET，MQCB，MQINQ 或 MQCLOSE 调用。

如果未指定 MQSO_MANAGED，那么无法在 *Hobj* 参数中提供从先前 MQSUB 调用返回的对象句柄。

MQSO_NO_多点广播

请求将发布发送到的目标不是多点广播组地址。仅当与 MQSO_MANAGED 选项结合使用时，此选项才有效。当 *Hobj* 参数中提供了队列的句柄时，多点广播不能用于此预订，并且该选项无效。

如果使用 MCAST(ONLY) 设置将主题定义为仅允许多点广播预订，那么调用将失败，原因码为 MQRC_MULTICAST_REQUIRED。

作用域选项: 以下选项控制正在进行的预订的作用域。如果使用 MQSO_ALTER 选项改变现有预订，那么无法更改此预订作用域选项。使用 MQSO-RESUME 从 MQSUB 调用返回时，将设置相应的作用域选项。

MQSO_SCOPE_QMGR

此预订仅在本地队列管理器上进行。没有代理预订分发到网络中的其他队列管理器。只有在此队列管理器上发布的发布才会发送到此订户。这将覆盖使用 SUBSCOPE 主题属性设置的任何行为。

注: 如果未设置，那么预订作用域由 SUBSCOPE 主题属性确定。

注册选项: 以下选项控制向此预订的队列管理器进行的注册的详细信息。如果使用 MQSO_ALTER 选项更改现有预订，那么可以更改这些注册选项。从使用 MQSO-RESUME 的 MQSUB 调用返回时，将设置相应的注册选项。

MQSO_GROUP_SUB

此预订将与使用同一队列并指定相同相关标识的同一 SubLevel 的其他预订一起分组，以便由于使用一组重叠的主题字符串，导致向预订组提供多个发布消息的主题的任何发布仅会导致将一条消息传递到队列。如果未使用此选项，那么将为匹配的每个唯一预订(由 SubName 标识)提供一个发布副本，这可能意味着可以将该发布的多个副本放在由多个预订共享的队列上。

只有该组中最重要的订阅才会提供该出版物的副本。最重要的预订基于完整主题名称，直到找到通配符为止。如果在组中混合使用通配符方案，那么只有通配符的位置很重要。建议您不要在共享同一队列的一组预订中组合不同的通配符方案。

创建新的分组预订时，它仍必须具有唯一的 SubName，但如果它与组中现有预订的完整主题名称匹配，那么调用将失败，并返回 MQRC_DUPLICATE_GROUP_SUB。

如果组中最重要的预订还指定 MQSO_NOT_OWN_PUBS，并且这是来自同一应用程序的发布，那么不会将任何发布传递到队列。

更改使用此选项进行的预订时，无法更改暗示分组的字段，MQSUB 调用上的 *Hobj* (表示队列和队列管理器名称) 以及 SubCorrel 标识。尝试更改它们会导致调用失败，并返回 MQRC_GROUPING_NOT_ALTERABLE。

此选项必须与具有未设置为 MQCI_NONE 的 SubCorrel 标识的 MQSO_SET_CORREL_ID 组合，并且不能与 MQSO_MANAGED 组合。

MQSO_ANY_USERID

指定 MQSO_ANY_USERID 时，订户的身份不会限制为单个用户标识。这允许任何用户在具有适当权限时更改或恢复预订。只有单个用户可以在任何时候拥有预订。尝试恢复使用另一个应用程序当前正在使用的预订会导致调用失败并返回 MQRC_SUBSCRIPTION_IN_USE。

要将此选项添加到现有预订，MQSUB 调用(使用 MQSO_ALTER)必须来自与原始预订本身相同的用户标识。

如果 MQSUB 调用引用了设置了 MQSO_ANY_USERID 的现有预订，并且用户标识与原始预订不同，那么仅当新用户标识有权预订主题时，调用才会成功。成功完成时，将在发布消息中设置新用户标识的情况下，将此订户的未来发布放到订户队列中。

请勿同时指定 MQSO_ANY_USERID 和 MQSO_FIXED_USERID。如果两者都未指定，那么缺省值为 MQSO_FIXED_USERID。

MQSO_FIXED_USERID

指定 MQSO_FIXED_USERID 时，只能通过最后一个用户标识来变更或恢复预订。如果未变更预订，那么是创建该预订的用户标识。

如果 MQSUB 动词引用设置了 MQSO_ANY_USERID 的现有预订，并使用 MQSO_ALTER 更改该预订以使用选项 MQSO_FIXED_USERID，那么该预订的用户标识现在固定在此新用户标识上。仅当新用户标识具有预订主题的权限时，调用才会成功。

如果记录为拥有预订的用户标识以外的用户标识尝试恢复或变更 MQSO_FIXED_USERID 预订，那么调用将失败，并返回 MQRC_IDENTITY_MATCH。可以使用 DISPLAY SBSTATUS 命令查看预订的拥有用户标识。

请勿同时指定 MQSO_ANY_USERID 和 MQSO_FIXED_USERID。如果两者都未指定，那么缺省值为 MQSO_FIXED_USERID。

发布选项: 以下选项控制将发布发送到此订户的方式。如果使用 MQSO_ALTER 选项改变现有预订，那么可以更改这些发布选项。

MQSO_NOT_OWN_PUBS

告知代理程序应用程序不希望看到其自己的任何发布。如果连接句柄相同，那么会将发布视为源自同一应用程序。从使用 MQSO_RESUME 的 MQSUB 调用返回时，将设置此选项 (如果适用)。

MQSO_NEW_publicATIONS_ONLY

创建此预订时，将不发送当前保留的发布内容，仅发送新发布内容。仅当指定了 MQSO_CREATE 时，此选项才适用。对预订的任何后续更改都不会改变发布流程，因此保留在主题上的任何发布都将已作为新发布发送给订户。

如果在不使用 MQSO_CREATE 的情况下指定此选项，那么调用将失败并返回 MQRC_OPTIONS_ERROR。从使用 MQSO_RESUME 的 MQSUB 调用返回时，即使使用此选项创建了预订，也不会设置此选项。

如果未使用此选项，那么会将先前保留的消息发送到提供的目标队列。如果此操作由于错误 (MQRC_RETAINED_MSG_Q_ERROR 或 MQRC_RETAINED_NOT_交付) 而失败，那么预订的创建将失败。

MQSO_PUBLICICATIONS_ON_REQUEST

设置此选项指示订户将在需要时专门请求信息。队列管理器不会向订户发送未经请求的消息。每次使用先前 MQSUB 调用中的 Hsub 句柄进行 MQSUBRQ 调用时，都会向订户发送保留的发布内容 (如果在主题中指定了通配符，那么可能会发送多个发布内容)。由于使用此选项的 MQSUB 调用，未发送任何发布。从使用 MQSO_RESUME 的 MQSUB 调用返回时，将设置此选项 (如果适用)。

此选项与大于 1 的 SubLevel 组合无效。

预读选项: 以下选项控制是否在应用程序发出请求之前将非持久消息发送到应用程序。

MQSO_READ_AHEAD_AS_Q_DEF

如果 MQSUB 调用使用受管句柄，那么与预订的主题相关联的模型队列的缺省预读属性将确定在应用程序请求消息之前是否将消息发送到应用程序。

这是缺省值。

MQSO_NO_READ_AHEAD

如果 MQSUB 调用使用受管句柄，那么在应用程序请求消息之前，不会将消息发送到应用程序。

MQSO_READ_AHEAD

如果 MQSUB 调用使用受管句柄，那么可能会在应用程序请求消息之前将其发送到应用程序。

注:

以下说明适用于预读选项:

1. 只能指定其中一个选项。如果同时指定了 MQSO_READ_AHEAD 和 MQSO_NO_READ_AHEAD，那么将返回原因码 MQRC_OPTIONS_ERROR。仅当指定了 MQSO_MANAGED 时，这些选项才适用。
2. 当传递先前已打开的队列时，它们不适用于 MQSUB。请求时可能未启用预读。第一个 MQGET 调用上使用的 MQGET 选项可能会阻止启用预读。此外，当客户机连接到不支持预读的队列管理器时，将禁用预读。如果应用程序未作为 WebSphere MQ 客户机运行，那么将忽略这些选项。

通配符选项: 以下选项控制如何在 MQSD 的 ObjectString 字段中提供的字符串中解释通配符。只能指定其中一个选项。如果使用 MQSO_ALTER 选项改变现有预订, 那么无法更改这些通配符选项。从使用 MQSO_RESUME 的 MQSUB 调用返回时, 将设置相应的通配符选项。

MQSO_WILDCARD_CHAR

通配符仅对主题字符串中的字符起作用。

MQSO_WILDCARD_CHAR 定义的行为显示在下表中。

| 特殊字符 | 行为 |
|---------|---|
| 正斜杠 (/) | 没有意义, 只是另一个角色 |
| 星号 (*) | 通配符, 零个或多个字符 |
| 问号 (?) | 通配符, 1 字符 |
| 百分号 (%) | 转义字符, 允许在字符串中使用字符 (*), (?) 或 (%), 而不解释为特殊字符, 例如 (% *), (%?) 或 (%%)。 |

例如, 发布以下主题:

```
/level0/level1/level2/level3/level4
```

与使用以下主题的订户匹配:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?e12/level3/level4
```

注: 使用通配符正好提供了 WebSphere MQ V6 和 WebSphere MB V6 中为发布/预订使用 MQRFH1 格式的消息时提供的含义。建议不要用于新编写的应用程序, 而仅用于先前针对该版本运行且未更改为使用缺省通配符行为的应用程序, 如 MQSO_WILDCARD_TOPIC 中所述。

MQSO_WILDCARD_TOPIC

通配符仅对主题字符串中的主题元素执行操作。如果未选择任何选项, 那么这是缺省行为。

MQSO_WILDCARD_TOPIC 所需的行为如下表所示:

| 特殊字符 | 行为 |
|------------|---|
| (/) | 主题级别分隔符 |
| 数字符号 (#) | 通配符: 多个主题级别 |
| 加号 (+) | 通配符: 单个主题级别 |
| 注意: | 如果 (+) 和 (#) 与主题级别中的其他字符 (包括自身) 混合在一起, 那么它们不会被视为通配符。在以下字符串中, (#) 和 (+) 字符被视为普通字符。 |
| | <pre>level0/level1/#+/level3/level#</pre> |

例如, 发布以下主题:

```
/level0/level1/level2/level3/level4
```

与使用以下主题的订户匹配:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

注: 当使用 MQRFH2 格式的消息进行发布/预订时, 通配符的使用提供了 WebSphere Message Broker Version 6 中提供的含义。

其他选项: 以下选项控制发出 API 调用而不是预订的方式。从使用 MQSO_RESUME 的 MQSUB 调用返回时, 这些选项保持不变。请参阅第 492 页的『AlternateUser 标识 (MQCHAR12)』以获取更多详细信息。

MQSO_ALTERNATE_USER_AUTHORITY

AlternateUser 标识字段包含用于验证此 MQSUB 调用的用户标识。仅当此 AlternateUser 标识有权使用指定的访问选项打开对象时, 无论运行应用程序的用户标识是否有权执行此操作, 该调用才能成功。

MQSO_SET_CORREL_ID

预订将使用 SubCorrelId 字段中提供的相关标识。如果未指定此选项, 那么队列管理器将在预订时自动创建相关标识, 并在 SubCorrelId 字段中返回到应用程序。有关更多信息, 请参阅第 502 页的『SubCorrel 标识 (MQBYTE24)』以获取更多详细信息。

此选项不能与 MQSO_MANAGED 组合使用。

MQSO_SET_IDENTITY_CONTEXT

预订将使用 PubAccountingToken 和 PubApplIdentityData 字段中提供的记帐令牌和应用程序身份数据。

如果指定了此选项, 那么将执行相同的授权检查, 就像使用带有 MQOO_SET_IDENTITY_CONTEXT 的 MQOPEN 调用访问目标队列一样, 但在同样使用 MQSO_MANAGED 选项的情况下, 目标队列上没有授权检查。

如果未指定此选项, 那么发送到此订户的发布具有与它们相关联的缺省上下文信息, 如下所示:

| MQMD 中的字段 | 使用的值 |
|------------------|--------------------------------------|
| UserIdentifier | 在进行预订时与预订关联的用户标识。 |
| AccountingToken | 如果可能, 请从环境确定; 如果可能, 请设置为 MQACT_NONE。 |
| ApplIdentityData | 设置为空白 |

此选项仅对 MQSO_CREATE 和 MQSO_ALTER 有效。如果与 MQSO_RESUME 配合使用, 那么将忽略 PubAccountingToken 和 PubApplIdentityData 字段, 因此此选项无效。

如果在不使用此选项的情况下更改了预订, 并且先前提供了预订的身份上下文信息, 那么将为已更改的预订生成缺省上下文信息。

如果允许不同用户标识将其与选项 MQSO_ANY_USERID 配合使用的预订由不同用户标识恢复, 那么将为现在拥有该预订的新用户标识生成缺省身份上下文, 并且将交付包含新身份上下文的任何后续发布。

MQSO_FAIL_IF QUIESCING

如果队列管理器处于停顿状态, 那么 MQSUB 调用将失败。在 z/OS 上, 对于 CICS 或 IMS 应用程序, 如果连接处于停顿状态, 那么此选项还会强制 MQSUB 调用失败。

PubAccounting 令牌 (MQBYTE32)

这是将位于与此预订匹配的所有发布消息的消息描述符 (MQMD) 的 AccountingToken 字段中的值。AccountingToken 是消息的身份上下文的一部分。有关消息上下文的更多信息, 请参阅消息上下文。有关 MQMD 中的 AccountingToken 字段的更多信息, 请参阅第 367 页的『AccountingToken (MQBYTE32)』

可以将以下特殊值用于 PubAccountingToken 字段:

MQACT_NONE

未指定记帐标记。

对于字段的长度, 该值为二进制零。

对于 C 编程语言，还定义了常量 MQACT_NONE_ARRAY; 此值与 MQACT_NONE 相同，但是字符数组而不是字符串。

如果未指定选项 MQSO_SET_IDENTITY_CONTEXT，那么队列管理器将生成记帐令牌作为缺省上下文信息，并且此字段是一个输出字段，其中包含将在为此预订发布的每条消息中设置的 *AccountingToken*。

如果指定了选项 MQSO_SET_IDENTITY_CONTEXT，那么用户将生成记帐令牌，并且此字段是一个输入字段，其中包含要在此预订的每个发布中设置的 *AccountingToken*。

此字段的长度由 MQ_ACCOUNTING_TOKEN_LENGTH 提供。此字段的初始值为 MQACT_NONE。

如果使用 MQSO_ALTER 选项更改现有预订，那么可以更改未来任何发布消息中 *AccountingToken* 的值。

从使用 MQSO_RESUME 的 MQSUB 调用返回时，此字段将设置为当前用于预订的 *AccountingToken*。

PubApplIdentityData (MQCHAR32)

这是与此预订匹配的所有发布消息的消息描述符 (MQMD) 的 *ApplIdentityData* 字段中的值。*ApplIdentityData* 是消息的身份上下文的一部分。有关消息上下文的更多信息，请参阅 [消息上下文](#)。有关 MQMD 中的 *ApplIdentityData* 字段的更多信息，请参阅 [第 369 页的『ApplIdentity 数据 \(MQCHAR32\)』](#)。

如果未指定选项 MQSO_SET_IDENTITY_CONTEXT，那么在为此预订发布的每条消息中设置的 *ApplIdentityData* 为空白，作为缺省上下文信息。

如果指定了选项 MQSO_SET_IDENTITY_CONTEXT，那么用户将生成 *PubApplIdentityData*，并且此字段是一个输入字段，其中包含要在此预订的每个发布中设置的 *ApplIdentityData*。

此字段的长度由 MQ_APPL_IDENTITY_DATA_LENGTH 给出。此字段的初始值是 C 中的空字符串，在其他编程语言中为 32 个空白字符。

如果使用 MQSO_ALTER 选项更改现有预订，那么可以更改任何未来发布消息的 *ApplIdentityData*。

从使用 MQSO_RESUME 的 MQSUB 调用返回时，此字段将设置为当前用于预订的 *ApplIdentityData*。

PubPriority (MQLONG)

这是将位于与此预订匹配的所有发布消息的消息描述符 (MQMD) 的 *Priority* 字段中的值。有关 MQMD 中 *Priority* 字段的更多信息，请参阅 [第 388 页的『Priority \(MQLONG\)』](#)。

此值必须大于或等于零；零是最低优先级。并且，还可以使用下列特殊值：

MQPRI_PRIORITY_AS_Q_DEF

如果在 MQSUB 调用的 *Hobj* 字段中提供了预订队列，并且该队列不是受管句柄，那么将从此队列的 *DefPriority* 属性获取消息的优先级。如果队列是集群队列，或者在队列名称解析路径中有多个定义，那么在将发布消息放入队列时将确定优先级，如 [第 388 页的『Priority \(MQLONG\)』](#) 所述。

如果 MQSUB 调用使用受管句柄，那么消息的优先级将从与预订的主题相关联的模型队列的 *DefPriority* 属性中获取。

MQPRI_PRIORITY_AS_PUBLISHED

消息的优先级是原始发布的优先级。这是字段的初始值。

如果使用 MQSO_ALTER 选项更改现有预订，那么可以更改任何未来发布消息的 *Priority*。

从使用 MQSO_RESUME 的 MQSUB 调用返回时，此字段将设置为用于预订的当前优先级。

ResObject 字符串 (MQCHARV)

这是队列管理器解析 *ObjectName* 中提供的名称后的长对象名称。

如果 *ObjectString* 中提供了长对象名，而 *ObjectName* 中未提供任何内容，那么此字段中返回的值与 *ObjectString* 中提供的值相同。

如果省略此字段(即 `ResObjectString.VSBufSize` 为零), 那么不会返回 `ResObjectString`, 但会在 `ResObjectString.VSLength` 中返回长度。如果长度比完整的 `ResObject` 字符串短, 那么它将被截断, 并返回在提供的长度中可以容纳的最右边的字符数。

如果未正确指定 `ResObjectString`, 那么根据如何使用 `MQCHARV` 结构的描述, 或者如果该结构超过最大长度, 那么调用将失败, 原因码为 `MQRC_RES_OBJECT_STRING_ERROR`。

SelectionString (MQCHARV)

这是用于提供在预订来自主题的消息时使用的选择条件的字符串。

如果提供了缓冲区, 并且 `VSBufSize` 中还有一个正缓冲区长度, 那么此可变长度字段将在使用 `MQSO_RESUME` 选项的 `MQSUB` 调用输出时返回。如果在调用上未提供缓冲区, 那么将在 `MQCHARV` 的 `VSLength` 字段中仅返回选择字符串的长度。如果提供的缓冲区小于返回字段所需的空間, 那么只有 `VSBufSize` 字节会返回到提供的缓冲区。

如果未正确指定 `SelectionString`, 那么根据如何使用第 263 页的『`MQCHARV-变量长度字符串`』结构的描述, 或者如果该结构超过最大长度, 那么调用将失败, 原因码为 `MQRC_SELECTION_STRING_ERROR`。

`选择器` 中描述了 `SelectionString` 用法。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQSD_STRUC_ID

预订描述符结构的标识。

对于 C 编程语言, 还定义了常量 `MQSD_STRUC_ID_ARRAY`; 此值与 `MQSD_STRUC_ID` 相同, 但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 `MQSD_STRUC_ID`。

SubCorrel 标识 (MQBYTE24)

此字段包含与此预订匹配的所有发布的公共相关标识。



注意: 只能在发布/预订集群中的队列管理器之间传递相关标识, 而不能在层次结构中的队列管理器之间传递该标识。

为与此预订匹配而发送的所有发布都在消息描述符中包含此相关标识。如果多个预订从同一队列获取其发布内容, 那么使用 `MQGET` (按相关标识) 仅允许获取特定预订的发布内容。此相关标识可以由队列管理器或用户生成。

如果未指定选项 `MQSO_SET_CORREL_ID`, 那么队列管理器将生成相关标识, 并且此字段是包含将在为此预订发布的每条消息中设置的相关标识的输出字段。生成的相关标识由 4 字节的产品标识 (ASCII 或 EBCDIC 中的 `AMQX` 或 `CSQM`) 组成, 后跟唯一字符串的特定于产品的实现。

如果指定了选项 `MQSO_SET_CORREL_ID`, 那么用户将生成相关标识, 并且此字段是包含要在此预订的每个发布中设置的相关标识的输入字段。在此情况下, 如果字段包含 `MQCI_NONE`, 那么在为此预订发布的每条消息中设置的相关标识是由消息的原始放置创建的相关标识。

如果指定了 `MQSO_GROUP_SUB` 选项, 并且指定的相关标识与正在使用同一队列和重叠主题字符串的现有分组预订相同, 那么将仅随该发布内容的副本一起提供该组中最重要的预订。

此字段的长度由 `MQ_CORREL_ID_LENGTH` 给出。此字段的初始值为 `MQCI_NONE`。

如果您正在使用 `MQSO_ALTER` 选项改变现有预订, 并且此字段是输入字段, 那么可以更改预订相关标识, 除非预订是分组预订, 即, 已使用选项 `MQSO_GROUP_SUB` 创建预订相关标识, 在这种情况下, 无法更改预订相关标识。

从使用 `MQSO_RESUME` 的 `MQSUB` 调用返回时, 此字段将设置为预订的当前相关标识。

SubExpiry (MQLONG)

这是预订到期后的时间 (以十分之一秒为单位)。在经过此时间间隔后, 没有更多发布将与此预订匹配。一旦预订到期, 将不再向队列发送发布内容。但是, 已经存在的出版物在任何方面都不受影响。 *SubExpiry* 对发布到期无任何影响。

可识别以下特殊值:

MQEI_UNLIMITED

预订具有无限的到期时间。

如果使用 MQSO_ALTER 选项改变现有预订, 那么可以更改预订的到期时间。

从使用 MQSO_RESUME 选项的 MQSUB 调用返回时, 此字段将设置为预订的原始到期时间, 而不是剩余的到期时间。

SubLevel (MQLONG)

这是与预订关联的级别。仅当发布位于具有小于或等于发布时使用的 PubLevel 的最高 SubLevel 值的预订集中时, 才会将其交付到此预订。但是, 如果保留了某个发布, 那么它将不再可供更高级别的订户使用, 因为它将在 PubLevel 1 上重新发布。

该值必须在 0 到 9 的范围内。0 是最低级别。

此字段的初始值为 1。

有关更多信息, 请参阅 [拦截发布](#)。

如果使用 MQSO_ALTER 选项改变现有预订, 那么无法更改 SubLevel。

不允许将值大于 1 的 SubLevel 与选项 MQSO_PUBLICATIONS_ON_REQUEST 组合。

从使用 MQSO_RESUME 的 MQSUB 调用返回时, 此字段将设置为用于预订的当前级别。

SubUser 数据 (MQCHARV)

这将指定预订用户数据。在此字段中为预订提供的数据将作为发送到此预订的每份发布内容的 MQSubUserData 消息属性。

SubUserData 的最大长度为 10240。

如果未正确指定 *SubUserData*, 那么根据如何使用 MQCHARV 结构的描述, 或者如果该结构超过最大长度, 那么调用将失败, 原因码为 MQRC_SUB_USER_DATA_ERROR。

这是一个输入字段。此结构中字段的初始值与 MQCHARV 结构中的初始值相同。

如果使用 MQSO_ALTER 选项改变现有预订, 那么可以更改预订用户数据。

如果提供了缓冲区并且 *VSubLen* 中存在正缓冲区长度, 那么将使用 MQSO_RESUME 选项在 MQSUB 调用的输出中返回此可变长度字段。如果在调用上未提供缓冲区, 那么仅会在 MQCHARV 的 *VSLength* 字段中返回预订用户日期的长度。如果提供的缓冲区小于返回字段所需的空間, 那么在提供的缓冲区中仅返回 *VSubLen* 字节。

SubName (MQCHARV)

这将指定预订名称。仅当 *Options* 指定了选项 MQSO_持久性时, 此字段才是必需的, 但如果提供的值也将由队列管理器用于 MQSO_NON_持久性。

如果指定了 *SubName*, 那么它在队列管理器中必须唯一, 因为它是用于标识预订的方法。

SubName 的最大长度为 10240。

此字段有两个用途。对于 MQSO_持久预订, 您可以使用此字段来标识预订, 以便在创建预订之后, 如果您已关闭预订的句柄 (使用 MQCO_KEEP_SUB 选项) 或者已与队列管理器断开连接, 那么可以恢复该预订。这是使用带有 MQSO_RESUME 选项的 MQSUB 调用完成的。它还显示在 DISPLAY SBSTATUS 的 SUBNAME 字段中的预订管理视图中。

如果未正确指定 *SubName*，那么根据如何使用 MQCHARV 结构的描述，在需要时将其排除在外 (即 *SubName*)。VSLength 为零)，或者如果它超过最大长度，那么调用将失败，原因码为 MQRC_SUB_NAME_ERROR。

这是一个输入字段。此结构中字段的初始值与 MQCHARV 结构中的初始值相同。

如果使用 MQSO_ALTER 选项更改现有预订，那么无法更改预订名称，因为它是用于查找所引用预订的标识字段。在使用 MQSO_RESUME 选项的 MQSUB 调用的输出中，不会对其进行更改。

Version (MQLONG)

这是结构版本号; 值必须为:

MQSD_VERSION_1

Version-1 预订描述符结构。

以下常量指定当前版本的版本号:

MQSD_CURRENT_VERSION

当前版本的预订描述符结构。

这始终是一个输入字段。此字段的初始值为 MQSD_VERSION_1。

使用主题字符串

根据主题对象中标识的子主题以及应用程序提供的子主题来构造主题。您可以使用子主题作为主题名称，也可以将其组合以形成新的主题名称。

在 MQI 程序中，完整主题名称由 MQOPEN 创建。它由发布/预订 MQI 调用中使用的两个字段组成，按列出的顺序:

1. 主题对象的 **TOPICSTR** 属性，在 **ObjectName** 字段中指定。
2. 定义应用程序提供的子主题的 **ObjectString** 参数。

生成的主题字符串将在 **ResObjectString** 参数中返回。

如果每个字段的第一个字符不是空白或空字符，并且字段长度大于零，那么这些字段被视为存在。如果仅存在其中一个字段，那么将其用作主题名称。如果两个字段都没有值，那么调用将失败，原因码为 MQRC_UNKNOWN_OBJECT_NAME 或 MQRC_TOPIC_STRING_ERROR (如果完整主题名称无效)。

如果这两个字段都存在，那么将在生成的组合主题名称的两个元素之间插入 "/" 字符。

第 504 页的表 548 显示了主题字符串并置的示例:

| TOPICSTR | ObjectString | 完整主题名称 | 注释 |
|----------|--------------|----------|-------------------|
| 足球/苏格兰人 | '' | 足球/苏格兰人 | TOPICSTR 单独使用 |
| '' | 足球/苏格兰人 | 足球/苏格兰人 | 单独使用 ObjectString |
| 美式足球 | 评分 | 足球/苏格兰人 | 在并置点添加 "/" 字符 |
| 美式足球 | /得分 | 足球//苏格兰人 | 在两个字符串之间生成 "空节点" |
| /足球 | 评分 | /足球/苏格兰人 | 主题以 "空节点" 开头 |

"/" 字符被视为特殊字符，用于向 主题树 中的完整主题名称提供结构，并且不得用于任何其他原因，因为主题树的结构受影响。主题 "/Football" 与主题 "Football" 不同。

以下通配符是特殊字符:

- 加号 "+"
- 编号符号 "#"
- 星号 "*"

- 问号 "?"

这些字符不会被视为无效，但是您必须确保了解这些字符的使用方式。发布时，您可能不希望在主题字符串中使用这些字符。可以使用通配符方案在主题级别中使用 "#" 或 "+" 与其他字符 (包括它们本身) 混合的主题字符串上进行发布。使用 "#" 或 "+" 作为两个 "/" 字符之间的唯一字符在主题字符串上发布会生成应用程序无法使用通配符方案 MQSO_WILDCARD_TOPIC 显式预订的主题字符串。这种情况会导致应用程序获得比预期更多的发布。

示例代码片段

从示例程序 [Example 2: Publisher to a variable topic](#) 中抽取的此代码片段将主题对象与变量主题字符串组合在一起。

```
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);
```

MQSD 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|----------------------------|-------------------------|-------------------|
| <i>StrucId</i> | MQSD_STRUC_ID | 'SD↵↵' |
| <i>Version</i> | MQSD_VERSION_1 | 1 |
| <i>Options</i> | MQSO_non_持久 | 0 |
| <i>ObjectName</i> | None | 空字符串或空白 |
| <i>AlternateUserId</i> | None | 空字符串或空白 |
| <i>AlternateSecurityId</i> | MQSID_NONE | Null |
| <i>SubExpiry</i> | MQEI_UNLIMITED | -1 |
| <i>ObjectString</i> | None | 为 MQCHARV 定义的名称和值 |
| <i>SubName</i> | None | 为 MQCHARV 定义的名称和值 |
| <i>SubUserData</i> | None | 为 MQCHARV 定义的名称和值 |
| <i>SubCorrelId</i> | MQCI_NONE | Null |
| <i>PubPriority</i> | MQPRI_PRIORITY_AS_Q_DEF | -3 |
| <i>PubAccountingToken</i> | MQACT_NONE | Null |
| <i>PubApplIdentityData</i> | None | 空字符串或空白 |
| <i>Selection String</i> | None | 为 MQCHARV 定义的名称和值 |
| <i>SubLevel</i> | None | 1 |
| <i>ResObjectString</i> | None | 为 MQCHARV 定义的名称和值 |

| 字段名称 | 常量的名称 | 常量值 |
|---|-------|-----|
| 注意: | | |
| 1. 符号 _ 表示单个空白字符。 | | |
| 2. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。 | | |
| 3. 在 C 编程语言中，宏变量 MQSD_DEFAULT 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值: | | |
| <pre>MQSD MySD = {MQSD_DEFAULT};</pre> | | |

C 声明

```
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 声明

```
** 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.

```

PL/I 声明

```

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,
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,
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 */

```

High Level Assembler 声明

| | | | |
|-----------------|-------|------|-------------------------------------|
| 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 | |
| MQSD_OBJECTSTRING_AREA | ORG | MQSD_OBJECTSTRING | |
| * | 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 | |
| MQSD_SUBNAME_AREA | ORG | MQSD_SUBNAME | |
| * | 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 | |
| MQSD_SUBUSERDATA_AREA | ORG | MQSD_SUBUSERDATA | |
| * | 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 | |
| MQSD_SELECTIONSTRING_AREA | ORG | MQSD_SELECTIONSTRING | |
| * | 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 | |
| MQSD_RESOBJECTSTRING_AREA | ORG | MQSD_RESOBJECTSTRING | |
| * | DS | CL(MQSD_RESOBJECTSTRING_LENGTH) | |
| MQSD_LENGTH | EQU | *-MQSD | |
| | ORG | MQSD | |
| MQSD_AREA | DS | CL(MQSD_LENGTH) | |

MQSMPO-设置消息属性选项

下表汇总了结构中的字段。

表 549: MQSMPO 中的字段

| 字段 | 描述 | Topic |
|----------------------|--------|-------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 选项 | 选项 |
| <i>ValueEncoding</i> | 属性值编码 | ValueEncoding |
| <i>ValueCCSID</i> | 属性值字符集 | ValueCCSID |

MQSMPO 概述

可用性: 所有 WebSphere MQ 系统和 WebSphere MQ 客户机。

用途: MQSMPO 结构允许应用程序指定用于控制如何设置消息属性的选项。该结构是 MQSETMP 调用上的输入参数。

字符集和编码: MQSMPO 中的数据必须位于应用程序的字符集和应用程序的编码 (MQENC_NATIVE) 中。

MQSMPO 的字段

MQSMPO 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

选项 (MQLONG)

位置选项: 下列选项与属性相对于属性光标的相对位置相关:

MQSMPO_SET_FIRST

设置与指定名称匹配的属性的值, 或者如果该属性不存在, 那么在具有匹配层次结构的所有其他属性之后添加新属性。

MQSMPO_SET_PROP_UNDER_CURSOR

设置属性光标指向的属性值。属性光标指向的属性是上次使用 MQIMPO_INQ_FIRST 或 MQIMPO_INQ_NEXT 选项查询的属性。

在 MQGET 调用上复用消息句柄时, 或者在 MQPUT 调用上的 MQGMO 或 MQPMO 结构的 *MsgHandle* 字段中指定消息句柄时, 将重置属性光标。

如果在尚未建立属性光标时使用此选项, 或者如果已删除属性光标指向的属性指针, 那么调用将失败并返回完成代码 MQCC_FAILED 和原因码 MQRC_PROPERTY_NOT_AVAILABLE。

MQSMPO_SET_PROP_BEFORE_CURSOR

在属性光标指向的属性之前设置新属性。属性光标指向的属性是上次使用 MQIMPO_INQ_FIRST 或 MQIMPO_INQ_NEXT 选项查询的属性。

在 MQGET 调用上复用消息句柄时, 或者在 MQPUT 调用上的 MQGMO 或 MQPMO 结构的 *MsgHandle* 字段中指定消息句柄时, 将重置属性光标。

如果在尚未建立属性光标时使用此选项, 或者如果已删除属性光标指向的属性指针, 那么调用将失败并返回完成代码 MQCC_FAILED 和原因码 MQRC_PROPERTY_NOT_AVAILABLE。

MQSMPO_SET_PROP_AFTER_CURSOR

在属性光标指向的属性之后设置新属性。属性光标指向的属性是上次使用 MQIMPO_INQ_FIRST 或 MQIMPO_INQ_NEXT 选项查询的属性。

在 MQGET 调用上复用消息句柄时, 或者在 MQPUT 调用上的 MQGMO 或 MQPMO 结构的 *MsgHandle* 字段中指定消息句柄时, 将重置属性光标。

如果在尚未建立属性光标时使用此选项, 或者如果已删除属性光标指向的属性指针, 那么调用将失败并返回完成代码 MQCC_FAILED 和原因码 MQRC_PROPERTY_NOT_AVAILABLE。

MQSMPO_APPEND_PROPERTY

导致在具有匹配层次结构的所有其他属性之后添加新属性。如果至少存在一个与指定名称匹配的属性，那么将在该属性列表末尾的末尾添加新属性。

此选项允许创建具有相同名称的属性的列表。

如果不需要所描述的任何选项，请使用以下选项：

MQSMPO_NONE

未指定任何选项。

这始终是一个输入字段。此字段的初始值为 `MQSMPO_SET_FIRST`。

StrucId (MQCHAR4)

这是结构标识；值必须为：

MQSMPO_STRUC_ID

设置消息属性选项结构的标识。

对于 C 编程语言，还定义了常量 `MQSMPO_STRUC_ID_ARRAY`；此值与 `MQSMPO_STRUC_ID` 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 `MQSMPO_STRUC_ID`。

ValueCCSID (MQLONG)

要设置的属性值的字符集（如果该值是字符串）。

这始终是一个输入字段。此字段的初始值为 `MQCCSI_APPL`。

ValueEncoding (MQLONG)

要设置的属性值的编码（如果该值是数字）。

这始终是一个输入字段。此字段的初始值为 `MQENC_NATIVE`。

Version (MQLONG)

这是结构版本号；值必须为：

MQSMPO_VERSION_1

Version-1 设置消息属性选项结构。

以下常量指定当前版本的版本号：

MQSMPO_CURRENT_VERSION

当前版本的设置消息属性选项结构。

这始终是一个输入字段。此字段的初始值为 `MQSMPO_VERSION_1`。

MQSMPO 的初始值和语言声明

| 表 550: MQSMPO 中字段的初始值 | | |
|-----------------------|-------------------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | <code>MQSMPO_STRUC_ID</code> | 'SMPO' |
| <i>Version</i> | <code>MQSMPO_VERSION_1</code> | 1 |
| <i>Options</i> | <code>MQSMPO_NONE</code> | 0 |

表 550: MQSMPO 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|----------------------|--------------|-------|
| <i>ValueEncoding</i> | MQENC_NATIVE | 取决于环境 |
| <i>ValueCCSID</i> | MQCCSI_APPL | -3 |

注意:

1. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。
2. 在 C 编程语言中，宏变量 MQSMPO_DEFAULT 包含上面列出的值。它可以通过以下方式用于为结构中的字段提供初始值:

```
MQSMPO MySMPO = {MQSMPO_DEFAULT};
```

C 声明

```
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 声明

```
** 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.
```

PL/I 声明

```
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 */
```

High Level Assembler 声明

```
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-预订请求选项

本部分描述了预订请求选项，其包含的字段以及这些字段的初始值。

| 字段 | 描述 | Topic |
|----------------|-------|-------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>Options</i> | 选项 | 选项 |
| <i>NumPubs</i> | 发布数 | NumPubs |

MQSRO 概述

可用性: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS 以及连接到这些系统的 WebSphere MQ MQI 客户机。

用途:MQSRO 结构允许应用程序指定用于控制如何发出预订请求的选项。此结构是 MQSUBRQ 调用上的输入/输出参数。

版本:MQSRO 的当前版本为 MQSRO_VERSION_1。

字符集和编码:MQSRO 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。但是，如果应用程序作为 MQ MQI 客户机运行，那么该结构必须采用客户机的字符集和编码。

MQSRO 的字段

MQSRO 结构包含以下字段; 这些字段按字母顺序进行描述:

NumPubs (MQLONG)

这是一个输出字段，返回到应用程序以指示由于此调用而发送到预订队列的发布数。虽然由于此调用而发送了此数目的发布，但无法保证此数目的消息可供应用程序获取，尤其是当它们是非持久消息时。

如果预订的主题包含通配符，那么可能有多个出版物。如果在创建由 *Hsub* 表示的预订时主题字符串中没有通配符，那么最多会由于此调用而发送一个发布内容。

选项 (MQLONG)

必须指定下列其中一个选项。只能指定一个选项。

MQSRO_FAIL_IF QUIESCING

如果队列管理器处于停顿状态，那么 MQSUBRQ 调用将失败。在 z/OS 上，对于 CICS 或 IMS 应用程序，如果连接处于停顿状态，那么此选项还会强制 MQSUBRQ 调用失败。

缺省选项: 如果不需要上述选项，那么必须使用以下选项:

MQSRO_NONE

使用此值来指示未指定任何其他选项; 所有选项均采用其缺省值。

MQSRO_NONE 可帮助程序文档。虽然不打算将此选项与任何其他选项一起使用，但由于其值为零，因此无法检测到此使用。

StrucId (MQCHAR4)

这是结构标识; 值必须为:

MQSRO_STRUC_ID

"预订请求选项" 结构的标识。

对于 C 编程语言，还定义了常量 MQSRO_STRUC_ID_ARRAY; 此值与 MQSRO_STRUC_ID 相同，但是字符数组而不是字符串。

这始终是一个输入字段。此字段的初始值为 MQSRO_STRUC_ID。

Version (MQLONG)

这是结构版本号; 值必须为:

MQSRO_VERSION_1

Version-1 预订请求选项结构。

以下常量指定当前版本的版本号:

MQSRO_CURRENT_VERSION

当前版本的 "预订请求选项" 结构。

这始终是一个输入字段。此字段的初始值为 MQSRO_VERSION_1。

MQSRO 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|----------------|-----------------|--------|
| <i>StrucId</i> | MQSRO_STRUC_ID | 'SRO↵' |
| <i>Version</i> | MQSRO_VERSION_1 | 1 |
| <i>Options</i> | MQSRO_NONE | 0 |
| <i>NumPubs</i> | None | 0 |

注意:

1. 符号 ↵ 表示单个空白字符。
2. 在 C 编程语言中，宏变量 MQSRO_DEFAULT 包含以上列出的值。它可以通过以下方式用于为结构中的字段提供初始值:

```
MQSRO MySRO = {MQSRO_DEFAULT};
```

C 声明

```
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 声明

```
** 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.
```

PL/I 声明

```
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 */

```

High Level Assembler 声明

```

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-状态报告结构

下表汇总了结构中的字段。

| 表 551: MQSTS 中的字段 | | |
|---------------------------------------|-----------------|-----------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>CompCode</i> | 第一个错误的完成代码 | CompCode |
| <i>Reason</i> | 第一个错误的原因码 | 原因 |
| <i>PutSuccessCount</i> | 成功的异步放置调用数 | SuccessCount |
| <i>PutWarningCount</i> | 具有警告的异步放置调用数 | WarningCount |
| <i>PutFailureCount</i> | 失败的异步放置调用数 | FailureCount |
| <i>ObjectType</i> | 失败对象的类型 | ObjectType |
| <i>ObjectName</i> | 失败对象的名称 | ObjectName |
| <i>ObjectQMgrName</i> | 拥有失败对象的队列管理器的名称 | ObjectQMgrName |
| <i>ResolvedObjectName</i> | 目标队列的已解析名称 | ResolvedObject 名称 |
| <i>ResolvedQMgrName</i> | 目标队列管理器的已解析名称 | ResolvedQMgrName |
| 注: 如果版本低于 MQSTS_VERSION_2, 那么将忽略其余字段。 | | |
| <i>ObjectString</i> | 失败对象的长对象名 | ObjectString |
| <i>SubName</i> | 失败预订的预订名称 | SubName |
| <i>OpenOptions</i> | 打开与故障关联的选项 | OpenOptions |
| <i>SubOptions</i> | 与故障关联的预订选项 | SubOptions |

MQSTS 概述

用途:MQSTS 结构是 MQSTAT 命令的输出参数。

字符集和编码: MQSTS 中的字符数据位于本地队列管理器的字符集中; 这是由 *CodedCharSetId* 队列管理器属性提供的。MQSTS 中的数字数据采用本机编码; 这是由 编码提供的。

用法:MQSTAT 命令用于检索状态信息。此信息在 MQSTS 结构中返回。有关 MQSTAT 的信息, 请参阅 [第 689 页的『MQSTAT-检索状态信息』](#)。

MQSTS 的字段

MQSTS 结构包含以下字段; 这些字段以 **字母顺序**描述:

CompCode (MQLONG)

正在报告的操作的完成代码。

CompCode 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

这是对 ObjectName 中指定的对象执行的先前异步放置操作所生成的完成代码。

MQSTAT_TYPE_RECONNECTION

如果连接正在重新连接或未能重新连接, 那么这是导致连接开始重新连接的完成代码。

如果连接当前已连接, 那么值为 MQCC_OK。

MQSTAT_TYPE_RECONNECTION_ERROR

如果连接未能重新连接, 那么这是导致重新连接失败的完成代码。

如果连接当前已连接或正在重新连接, 那么值为 MQCC_OK。

CompCode 始终是输出字段。其初始值为 MQCC_OK。

ObjectName (MQCHAR48)

要报告的对象名称。

ObjectName 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

这是 put 操作中使用的队列或主题的名称, 在 MQSTS 结构的 *CompCode* 和 *Reason* 字段中报告了该队列或主题的故障。

MQSTAT_TYPE_RECONNECTION

如果连接正在重新连接, 那么这是与连接关联的队列管理器的名称。

MQSTAT_TYPE_RECONNECTION_ERROR

如果连接未能重新连接, 那么这是导致重新连接失败的对象的名称。在 MQSTS 结构的 *CompCode* 和 *Reason* 字段中报告失败原因。

ObjectName 是输出字段。其初始值为 C 中的空字符串, 在其他编程语言中为 48 个空白字符。

ObjectQMgr 名称 (MQCHAR48)

要报告的队列管理器的名称。

ObjectQMgrName 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

这是定义了 *ObjectName* 对象的队列管理器的名称。名称完全为空白, 直到第一个空字符或字段的末尾表示应用程序所连接的队列管理器 (本地队列管理器)。

MQSTAT_TYPE_RECONNECTION

空白。

MQSTAT_TYPE_RECONNECTION_ERROR

如果连接未能重新连接, 那么这是导致重新连接失败的对象的名称。在 MQSTS 结构的 *CompCode* 和 *Reason* 字段中报告失败原因。

ObjectQMgrName 是输出字段。其值为 C 中的空字符串, 在其他编程语言中为 48 个空白字符。

ObjectString (MQCHARV)

要报告的失败对象的长对象名。仅在 MQSTS V 2 或更高版本中存在。

ObjectString 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

这是 MQPUT 操作中使用的队列或主题的长对象名，此操作失败。

MQSTAT_TYPE_RECONNECTION

零长度字符串

MQSTAT_TYPE_RECONNECTION_ERROR

这是导致重新连接失败的对象的长对象名。

ObjectString 是输出字段。其初始值为零长度字符串。

ObjectType (MQLONG)

要报告的 *ObjectName* 中指定的对象的类型。

第 144 页的『MQOT_* (对象类型和扩展对象类型)』中列出了 *ObjectType* 的可能值。

ObjectType 是输出字段。其初始值为 MQOT_Q。

OpenOptions (MQLONG)

用于打开要报告的对象的 *OpenOptions*。仅在 MQSTS V 2 或更高版本中存在。

OpenOptions 的值取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

零。

MQSTAT_TYPE_RECONNECTION

零。

MQSTAT_TYPE_RECONNECTION_ERROR

发生故障时使用的 *OpenOptions*。在 MQSTS 结构的 *CompCode* 和 *Reason* 字段中报告失败原因。

OpenOptions 是输出字段。其初始值为零。

PutFailure 计数 (MQLONG)

失败的异步放置操作数。

PutFailureCount 的值取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

使用 MQCC_FAILED 完成的对 MQSTS 结构中指定的对象的异步放置操作数。

MQSTAT_TYPE_RECONNECTION

零。

MQSTAT_TYPE_RECONNECTION_ERROR

零。

PutFailureCount 是输出字段。其初始值为零。

PutSuccess 计数 (MQLONG)

成功的异步放置操作数。

PutSuccessCount 的值取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

使用 MQCC_OK 完成的对 MQSTS 结构中指定的对象的异步放置操作数。

MQSTAT_TYPE_RECONNECTION

零。

MQSTAT_TYPE_RECONNECTION_ERROR

零。

PutSuccessCount 是输出字段。其初始值为零。

PutWarning 计数 (MQLONG)

以警告结束的异步放置操作数。

PutWarningCount 的值取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

使用 MQCC_WARNING 完成的对 MQSTS 结构中指定的对象的异步放置操作数。

MQSTAT_TYPE_RECONNECTION

零。

MQSTAT_TYPE_RECONNECTION_ERROR

零。

PutWarningCount 是输出字段。其初始值为零。

SubName (MQCHARV)

失败预订的名称。仅在 MQSTS V 2 或更高版本中存在。

SubName 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

零长度字符串。

MQSTAT_TYPE_RECONNECTION

零长度字符串。

MQSTAT_TYPE_RECONNECTION_ERROR

导致重新连接失败的预订的名称。如果没有可用的预订名称，或者故障与预订无关，那么这是长度为零的字符串。

SubName 是输出字段。其初始值为零长度字符串。

SubOptions (MQLONG)

用于打开失败预订的 SubOptions。仅在 MQSTS V 2 或更高版本中存在。

SubOptions 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

零。

MQSTAT_TYPE_RECONNECTION

零。

MQSTAT_TYPE_RECONNECTION_ERROR

发生故障时使用的 SubOptions。如果失败与预订主题无关，那么返回的值为零。

SubOptions 是输出字段。其初始值为零。

原因 (MQLONG)

要报告的操作的原因码。

Reason 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

这是对 ObjectName 中指定的对象执行先前异步放置操作所产生的原因码。

MQSTAT_TYPE_RECONNECTION

如果连接正在重新连接或未能重新连接，那么这是导致重新连接开始重新连接的原因码。

如果连接当前已连接，那么值为 MQRC_NONE。

MQSTAT_TYPE_RECONNECTION_ERROR

如果连接未能重新连接，那么这是导致重新连接失败的原因码。

如果连接当前已连接或正在重新连接，那么值为 MQRC_NONE。

Reason 是输出字段。其初始值为 MQRC_NONE。

ResolvedObject 名称 (MQCHAR48)

在本地队列管理器解析名称之后，在 *ObjectName* 中指定的对象的名称。

ResolvedObjectName 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

ResolvedObjectName 是在本地队列管理器解析名称后在 *ObjectName* 中指定的对象的名称。返回的名称是由 *ResolvedQMgrName* 标识的队列管理器上存在的对象的名称。

MQSTAT_TYPE_RECONNECTION

空白。

MQSTAT_TYPE_RECONNECTION_ERROR

空白。

ResolvedObjectName 是输出字段。其初始值为 C 中的空字符串，在其他编程语言中为 48 个空白字符。

ResolvedQMgr 名称 (MQCHAR48)

本地队列管理器解析名称后的目标队列管理器的名称。

ResolvedQMgrName 的解释取决于 MQSTAT Type 参数的值。

MQSTAT_TYPE_ASYNC_ERROR

ResolvedQMgrName 是本地队列管理器解析名称后的目标队列管理器的名称。返回的名称是拥有由 *ResolvedObjectName* 标识的对象的队列管理器的名称。*ResolvedQMgrName* 可能是本地队列管理器的名称。

MQSTAT_TYPE_RECONNECTION

空白。

MQSTAT_TYPE_RECONNECTION_ERROR

空白。

ResolvedQMgrName 始终是输出字段。其初始值为 C 中的空字符串，在其他编程语言中为 48 个空白字符。

StrucId (MQCHAR4)

状态报告结构 MQSTS 的标识。

StrucId 是结构标识。该值必须为：

MQSTS_STRUC_ID

状态报告结构的标识。

对于 C 编程语言，还定义了常量 MQSTS_STRUC_ID_ARRAY；此值与 MQSTS_STRUC_ID 相同，但是字符数组而不是字符串。

StrucId 始终是输入字段。其初始值为 MQSTS_STRUC_ID。

Version (MQLONG)

结构版本号。

值必须为:

MQSTS_VERSION_1

V 1 状态报告结构。

MQSTS_VERSION_2

V 2 状态报告结构。

以下常量指定当前版本的版本号:

MQSTS_CURRENT_VERSION

当前版本的状态报告结构。当前版本为 MQSTS_VERSION_2。

Version 始终是输入字段。其初始值为 MQSTS_VERSION_1。

MQSTS 的初始值和语言声明

| 表 552: MQSTS 中字段的初始值 | | |
|--|-----------------|-----------------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQSTS_STRUC_ID | 'STAT↵' |
| <i>Version</i> | MQSTS_VERSION_1 | 1 |
| <i>CompCode</i> | MQCC_OK | 0 |
| <i>Reason</i> | MQRC_NONE | 0 |
| <i>PutSuccessCount</i> | None | 0 |
| <i>PutWarningCount</i> | None | 0 |
| <i>PutFailureCount</i> | None | 0 |
| <i>ObjectType</i> | MQOT_Q | 1 |
| <i>ObjectName</i> | None | 空字符串或空白 |
| <i>ObjectQMgrName</i> | None | 空字符串或空白 |
| <i>ResolvedObjectName</i> | None | 空字符串或空白 |
| <i>ResolvedQMgrName</i> | None | 空字符串或空白 |
| <i>ObjectString</i> | MQCHARV_DEFAULT | {NULL,0,0,0,-3} |
| <i>SubName</i> | MQCHARV_DEFAULT | {NULL,0,0,0,-3} |
| <i>OpenOptions</i> | None | 0 |
| <i>SubOptions</i> | None | 0 |
| <p>注意:</p> <ol style="list-style-type: none"> 1. 符号 ↵ 表示单个空白字符。 2. 值 Null 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。 3. 在 C 编程语言中，宏变量 MQSTS_DEFAULT 包含上面列出的值。它可以通过以下方式用于为结构中的字段提供初始值: <pre>MQSTS MySTS = {MQSTS_DEFAULT};</pre> | | |

C 声明

```
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 声明

```
** 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 **

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

| | | |
|------------------------------|-------|-------------------------------------|
| 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 |
| MQSTS_OBJECTSTRING_AREA | DS | ORG MQSTS_OBJECTSTRING |
| * | | 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 |
| MQSTS_SUBNAME_AREA | DS | ORG MQSTS_SUBNAME |
| * | | CL(MQSTS_SUBNAME_LENGTH) |
| MQSTS_OPENOPTIONS | DS | F Failing open options |
| MQSTS_SUBOPTIONS | DS | F Failing subscription option |
| MQSTS_LENGTH | EQU | *-MQSTS |
| MQSTS_AREA | DS | ORG MQSTS |
| | | CL(MQSTS_LENGTH) |

MQTM-触发器消息

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|--------------------|----------|-----------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>QName</i> | 已触发队列的名称 | QName |
| <i>ProcessName</i> | 进程对象的名称 | ProcessName |
| <i>TriggerData</i> | 触发器数据 | TriggerData |
| <i>ApplType</i> | 应用程序类型 | ApplType |
| <i>ApplId</i> | 应用程序标识 | ApplId |
| <i>EnvData</i> | 环境数据 | EnvData |
| <i>UserData</i> | 用户数据 | UserData |

MQTM 概述

用途:MQTM 结构描述当队列发生触发器事件时，队列管理器发送到触发器监视器应用程序的触发器消息中的数据。

此结构是 WebSphere MQ 触发器监视器接口 (TMI) 的一部分，TMI 是 WebSphere MQ 框架接口之一。

格式名:MQFMT_TRIGGER。

字符集和编码:MQTM 中的字符数据位于生成 MQTM 的队列管理器的字符集中。MQTM 中的数字数据采用生成 MQTM 的队列管理器的机器编码。

MQTM 的字符集和编码由以下内容中的 *CodedCharSetId* 和 *Encoding* 字段提供:

- MQMD (如果 MQTM 结构位于消息数据的开头)，或者
- MQTM 结构之前的头结构 (所有其他情况)。

用法: 触发器监视器应用程序可能需要将触发器消息中的部分或全部信息传递到触发器监视器应用程序启动的应用程序。启动的应用程序可能需要的信息包括 *QName*、*TriggerData* 和 *UserData*。触发器监视器应用程序可以将 MQTM 结构直接传递给已启动的应用程序，也可以改为传递 MQTMC2 结构，这取决于环境允许的内容以及启动的应用程序的方便程度。有关 MQTMC2 的信息，请参阅第 528 页的『MQTMC2 -触发器消息 2 (字符格式)』。

- 在 z/OS 上，对于使用 CKTI 事务启动的 MQAT_CICS 应用程序，整个触发器消息结构 MQTM 可供启动的事务使用; 可使用 EXEC CICS RETRIEVE 命令检索信息。
- 在 IBM i 上，随 WebSphere MQ 提供的触发器监视器应用程序将 MQTMC2 结构传递到已启动的应用程序。

有关使用触发器的信息，请参阅 [使用触发器启动 WebSphere MQ 应用程序](#)。

触发器消息的 MQMD: 队列管理器生成的触发器消息的 MQMD 中的字段设置如下:

| MQMD 中的字段 | 使用的值 |
|----------------|----------------|
| <i>StrucId</i> | MQMD_STRUC_ID |
| <i>Version</i> | MQMD_VERSION_1 |
| <i>Report</i> | MQRO_NONE |
| <i>MsgType</i> | MQMT_DATAGRAM |
| <i>Expiry</i> | MQEI_UNLIMITED |

| MQMD 中的字段 | 使用的值 |
|-------------------------|---------------------------------|
| <i>Feedback</i> | MQFB_NONE |
| <i>Encoding</i> | MQENC_NATIVE |
| <i>CodedCharSetId</i> | 队列管理器的 <i>CodedCharSetId</i> 属性 |
| <i>Format</i> | MQFMT_TRIGGER |
| <i>Priority</i> | 启动队列的 <i>DefPriority</i> 属性 |
| <i>Persistence</i> | MQPER_NOT_PERSISTENT |
| <i>MsgId</i> | 唯一值 |
| <i>CorrelId</i> | MQCI_NONE |
| <i>BackoutCount</i> | 0 |
| <i>ReplyToQ</i> | 空白 |
| <i>ReplyToQMGr</i> | 队列管理器的名称 |
| <i>UserIdentifier</i> | 空白 |
| <i>AccountingToken</i> | MQACT_NONE |
| <i>ApplIdentityData</i> | 空白 |
| <i>PutApplType</i> | MQAT_QMGR, 或者适用于消息通道代理程序 |
| <i>PutApplName</i> | 队列管理器名称的前 28 个字节 |
| <i>PutDate</i> | 发送触发器消息的日期 |
| <i>PutTime</i> | 发送触发器消息的时间 |
| <i>ApplOriginData</i> | 空白 |

建议生成触发器消息的应用程序设置类似的值, 但以下值除外:

- 可以将 *Priority* 字段设置为 MQPRI_PRIORITY_AS_Q_DEF (放入消息时, 队列管理器会将其更改为启动队列的缺省优先级)。
- 可以将 *ReplyToQMGr* 字段设置为空白 (队列管理器将在放入消息时将其更改为本地队列管理器的名称)。
- 根据需要为应用程序设置上下文字段。

MQTM 的字段

MQTM 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

ApplId (MQCHAR256)

这是一个字符串, 用于标识要启动的应用程序, 并由接收触发器消息的触发器监视器应用程序使用。队列管理器使用由 *ProcessName* 字段标识的流程对象的 *ApplId* 属性值来初始化此字段; 请参阅 [第 761 页的『进程定义的属性』](#) 以获取此属性的详细信息。此数据的内容对队列管理器没有任何意义。

ApplId 的含义由 trigger-monitor 应用程序确定。WebSphere MQ 提供的触发器监视器要求 *ApplId* 是可执行程序的名称。以下说明适用于所指示的环境:

- 在 z/OS 上, *ApplId* 为:
 - 使用 CICS 触发器监视器事务 CKTI 启动的应用程序的 CICS 事务标识
 - 使用 IMS 触发器监视器 CSQQTRMN 启动的应用程序的 IMS 事务标识
- 在 Windows 系统上, 可以使用驱动器和目录路径作为程序名的前缀。
- 在 IBM i 上, 可以使用库名和/或字符作为程序名的前缀。
- 在 UNIX 系统上, 可以使用目录路径作为程序名的前缀。

此字段的长度由 MQ_PROCESS_APPL_ID_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 256 个空白字符。

ApplType (MQLONG)

这标识要启动的程序的性质，并由接收触发器消息的触发器监视器应用程序使用。队列管理器使用由 *ProcessName* 字段标识的流程对象的 *ApplType* 属性值来初始化此字段；请参阅第 761 页的『进程定义的属性』以获取此属性的详细信息。此数据的内容对队列管理器没有任何意义。

ApplType 可以具有下列其中一个标准值。也可以使用用户定义的类型，但应该限制为 MQAT_USER_FIRST 到 MQAT_USER_LAST 范围内的值：

MQAT_AIX

AIX 应用程序 (与 MQAT_UNIX 的值相同)。

MQAT_BATCH

批处理应用程序

MQAT_BROKER

代理应用程序

MQAT_CICS

CICS 事务。

MQAT_CICS_BRIDGE

CICS 网桥应用程序。

MQAT_CICS_VSE

CICS/VSE 事务。

MQAT_DOS

PC DOS 上的 WebSphere MQ MQI 客户机应用程序。

MQAT_IMS

IMS 应用程序。

MQAT_IMS_BRIDGE

IMS 网桥应用程序。

MQAT_JAVA

Java 应用程序。

MQAT_MVS

MVS 或 TSO 应用程序 (与 MQAT_ZOS 值相同)。

MQAT_NOTES_AGENT

Lotus Notes 代理程序应用程序。

MQAT_NSK

HP Integrity NonStop Server 应用程序。

MQAT_OS390

OS/390 应用程序 (与 MQAT_ZOS 的值相同)。

MQAT_OS400

IBM i 应用程序。

MQAT_RRS_BATCH

RRS 批处理应用程序。

MQAT_UNIX

UNIX 应用程序。

MQAT_UNKNOWN

未知类型的应用程序。

MQAT_USER

用户定义的应用程序类型。

MQAT_VOS

Stratus VOS 应用程序。

MQAT_WINDOWS

16 位 Windows 应用程序。

MQAT_WINDOWS_NT

32 位 Windows 应用程序。

MQAT_WLM

z/OS 工作负载管理器应用程序。

MQAT_XCF

XCF。

MQAT_ZOS

z/OS 应用程序。

MQAT_USER_FIRST

用户定义的应用程序类型的最小值。

MQAT_USER_LAST

用户定义的应用程序类型的最大值。

此字段的初始值为 0。

EnvData (MQCHAR128)

这是一个字符串，其中包含与要启动的应用程序相关的环境信息，并且由接收触发器消息的触发器监视器应用程序使用。队列管理器使用由 *ProcessName* 字段标识的流程对象的 *EnvData* 属性值来初始化此字段；请参阅 [第 761 页的『进程定义的属性』](#) 以获取此属性的详细信息。此数据的内容对队列管理器没有任何意义。

在 z/OS 上，对于使用 CKTI 事务启动的 CICS 应用程序或要使用 CSQQTRMN 事务启动的 IMS 应用程序，不使用此信息。

此字段的长度由 MQ_PROCESS_ENV_DATA_LENGTH 指定。此字段的初始值是 C 中的空字符串，在其他编程语言中为 128 个空白字符。

ProcessName (MQCHAR48)

这是为触发的队列指定的队列管理器进程对象的名称，可由接收触发器消息的触发器监视器应用程序使用。队列管理器使用 *QName* 字段标识的队列的 *ProcessName* 属性值来初始化此字段；请参阅 [第 731 页的『队列的属性』](#) 以获取此属性的详细信息。

比定义的字段长度短的名称总是用空格填充到右边；它们不会被空字符过早结束。

此字段的长度由 MQ_PROCESS_NAME_LENGTH 指定。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

QName (MQCHAR48)

这是发生触发器事件的队列的名称，由触发器监视器应用程序启动的应用程序使用。队列管理器使用触发队列的 *QName* 属性的值来初始化此字段；请参阅 [第 731 页的『队列的属性』](#) 以获取此属性的详细信息。

比定义的字段长度短的名称将用空格填充到右边；它们不会以空字符过早结束。

此字段的长度由 MQ_Q_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串，在其他编程语言中为 48 个空白字符。

StrucId (MQCHAR4)

这是结构标识。该值必须为：

MQTM_STRUC_ID

触发器消息结构的标识。

对于 C 编程语言，还定义了常量 MQTM_STRUC_ID_ARRAY；此值与 MQTM_STRUC_ID 相同，但是字符数组而不是字符串。

此字段的初始值为 MQTM_STRUC_ID。

TriggerData (MQCHAR64)

这是自由格式数据，供接收触发器消息的触发器监视器应用程序使用。队列管理器使用 *QName* 字段标识的队列的 *TriggerData* 属性值来初始化此字段；请参阅第 731 页的『队列的属性』以获取此属性的详细信息。此数据的内容对队列管理器没有任何意义。

在 z/OS 上，对于使用 CKTI 事务启动的 CICS 应用程序，不使用此信息。

此字段的长度由 MQ_TRIGGER_DATA_LENGTH 指定。此字段的初始值是 C 中的空字符串，在其他编程语言中为 64 个空白字符。

UserData (MQCHAR128)

这是一个字符串，其中包含与要启动的应用程序相关的用户信息，并且由接收触发器消息的触发器监视器应用程序使用。队列管理器使用由 *ProcessName* 字段标识的流程对象的 *UserData* 属性值来初始化此字段；请参阅第 761 页的『进程定义的属性』以获取此属性的详细信息。此数据的内容对队列管理器没有任何意义。

对于 Microsoft Windows，如果要将进程定义传递到 `runmqtrm`，那么字符串不得包含双引号。

此字段的长度由 MQ_PROCESS_USER_DATA_LENGTH 给出。此字段的初始值是 C 中的空字符串，在其他编程语言中为 128 个空白字符。

Version (MQLONG)

这是结构版本号。该值必须为：

MQTM_VERSION_1

触发器消息结构的版本号。

以下常量指定当前版本的版本号：

MQTM_CURRENT_VERSION

触发器消息结构的当前版本。

此字段的初始值为 MQTM_VERSION_1。

MQTM 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|--------------------|----------------|---------|
| <i>StrucId</i> | MQTM_STRUC_ID | 'TM--' |
| <i>Version</i> | MQTM_VERSION_1 | 1 |
| <i>QName</i> | None | 空字符串或空白 |
| <i>ProcessName</i> | None | 空字符串或空白 |
| <i>TriggerData</i> | None | 空字符串或空白 |
| <i>ApplType</i> | None | 0 |
| <i>ApplId</i> | None | 空字符串或空白 |
| <i>EnvData</i> | None | 空字符串或空白 |
| <i>UserData</i> | None | 空字符串或空白 |

表 554: MQTM 的 MQTM 中字段的初始值 (继续)

| 字段名称 | 常量的名称 | 常量值 |
|---|-------|-----|
| 注意: | | |
| <ol style="list-style-type: none"> 1. 符号 <code>\</code> 表示单个空白字符。 2. 值 <code>Null</code> 字符串或空白表示 C 中的空字符串，而空白字符表示其他编程语言中的空字符。 3. 在 C 编程语言中，宏变量 <code>MQTM_DEFAULT</code> 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值: | | |
| <pre>MQTM MyTM = {MQTM_DEFAULT};</pre> | | |

C 声明

```
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 声明

```
** 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).
```

PL/I 声明

```
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 */
```

High Level Assembler 声明

```

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)

```

Visual Basic 声明

```

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 - 触发器消息 2 (字符格式)

下表汇总了结构中的字段。

| 表 555: MQTMC2 中的字段 | | |
|--------------------|----------|-----------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>QName</i> | 已触发队列的名称 | QName |
| <i>ProcessName</i> | 进程对象的名称 | ProcessName |
| <i>TriggerData</i> | 触发器数据 | TriggerData |
| <i>ApplType</i> | 应用程序类型 | ApplType |
| <i>ApplId</i> | 应用程序标识 | ApplId |
| <i>EnvData</i> | 环境数据 | EnvData |
| <i>UserData</i> | 用户数据 | UserData |
| <i>QMgrName</i> | 队列管理器名称 | QMgrName |

MQTMC2 概述

用途: 当触发器监视器应用程序从启动队列中检索触发器消息 (MQTM) 时，触发器监视器可能需要将触发器消息中的部分或全部信息传递到触发器监视器启动的应用程序。

启动的应用程序可能需要的信息包括 *QName*、*TriggerData* 和 *UserData*。触发器监视器应用程序可以将 MQTM 结构直接传递到已启动的应用程序，或者改为传递 MQTMC2 结构，这取决于环境允许的内容以及对已启动的应用程序的方便程度。

此结构是 WebSphere MQ 触发器监视器接口 (TMI) 的一部分，TMI 是 WebSphere MQ 框架接口之一。

字符集和编码: MQTMC2 中的字符数据位于本地队列管理器的字符集中; 这是由 *CodedCharSetId* 队列管理器属性提供的。

用法: MQTMC2 结构与 MQTM 结构的格式非常相似。不同的是, MQTM 中的非字符字段在 MQTMC2 中更改为相同长度的字符字段, 并且在结构末尾添加队列管理器名称。

- 在 z/OS 上, 对于使用 CSQQTRMN 应用程序启动的 MQAT_IMS 应用程序, MQTMC2 结构可供已启动的应用程序使用。
- 在 IBM i 上, 随 WebSphere MQ 提供的触发器监视器应用程序将 MQTMC2 结构传递到已启动的应用程序。

MQTMC2 的字段

MQTMC2 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

ApplId (MQCHAR256)

应用程序标识。

请参阅 MQTM 结构中的 *ApplId* 字段。

ApplType (MQCHAR4)

应用程序类型。

此字段始终包含空白, 无论原始触发器消息的 MQTM 结构中 *ApplType* 字段中的值是什么。

EnvData (MQCHAR128)

环境数据。

请参阅 MQTM 结构中的 *EnvData* 字段。

ProcessName (MQCHAR48)

进程对象的名称。

请参阅 MQTM 结构中的 *ProcessName* 字段。

QMgrName (MQCHAR48)

队列管理器名称。

这是发生触发器事件的队列管理器的名称。

QName (MQCHAR48)

触发队列的名称。

请参阅 MQTM 结构中的 *QName* 字段。

StrucId (MQCHAR4)

结构标识。

该值必须为:

MQTMC_STRUC_ID

触发器消息 (字符格式) 结构的标识。

对于 C 编程语言, 还定义了常量 MQTMC_STRUC_ID_ARRAY; 此值与 MQTMC_STRUC_ID 相同, 但是字符数组而不是字符串。

TriggerData (MQCHAR64)

触发器数据。

请参阅 MQTM 结构中的 *TriggerData* 字段。

UserData (MQCHAR128)

用户数据。

请参阅 MQTM 结构中的 *UserData* 字段。

版本 (MQCHAR4)

结构版本号。

该值必须为:

MQTMC_VERSION_2

V 2 触发器消息 (字符格式) 结构。

对于 C 编程语言, 还定义了常量 MQTMC_VERSION_2_ARRAY; 此值与 MQTMC_VERSION_2 相同, 但是字符数组而不是字符串。

以下常量指定当前版本的版本号:

MQTMC_CURRENT_VERSION

当前版本的触发器消息 (字符格式) 结构。

MQTMC2 的初始值和语言声明

| 字段名称 | 常量的名称 | 常量值 |
|--------------------|-----------------|---------|
| <i>StrucId</i> | MQTMC_STRUC_ID | 'TMC~' |
| <i>Version</i> | MQTMC_VERSION_2 | '~~2' |
| <i>QName</i> | None | 空字符串或空白 |
| <i>ProcessName</i> | None | 空字符串或空白 |
| <i>TriggerData</i> | None | 空字符串或空白 |
| <i>ApplType</i> | None | 空白 |
| <i>ApplId</i> | None | 空字符串或空白 |
| <i>EnvData</i> | None | 空字符串或空白 |
| <i>UserData</i> | None | 空字符串或空白 |
| <i>QMgrName</i> | None | 空字符串或空白 |

注意:

1. 符号 ~ 表示单个空白字符。
2. 值 Null 字符串或空白表示 C 中的空字符串, 而空白字符表示其他编程语言中的空字符。
3. 在 C 编程语言中, 宏变量 MQTMC2_DEFAULT 包含上面列出的值。通过以下方式使用它为结构中的字段提供初始值:

```
MQTMC2 MyTMC = {MQTMC2_DEFAULT};
```

C 声明

```
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 声明

```

** 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).

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

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)

```

Visual Basic 声明

```

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 - 工作信息头

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|-----------------------|------------------|--------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>StrucLength</i> | MQWIH 结构的长度 | StrucLength |
| <i>Encoding</i> | MQWIH 之后数据的数字编码 | 编码 |
| <i>CodedCharSetId</i> | MQWIH 之后数据的字符集标识 | CodedCharSetId |
| <i>Format</i> | MQWIH 之后数据的格式名称 | FORMAT |
| <i>Flags</i> | 标志 | 标志 |
| <i>ServiceName</i> | 服务名称 | ServiceName |
| <i>ServiceStep</i> | 服务步骤名称 | ServiceStep |
| <i>MsgToken</i> | 消息令牌 | MsgToken |
| <i>Reserved</i> | 已保留 | 已保留 |

MQWIH 概述

可用性: 所有 WebSphere MQ 系统，以及连接到这些系统的 WebSphere MQ 客户机。

用途: MQWIH 结构描述了要由 z/OS 工作负载管理器处理的消息开始时必须存在的信息。

格式名: MQFMT_WORK_INFO_HEADER。

字符集和编码: MQWIH 结构中的字段是由 MQWIH 之前的头结构中的 *CodedCharSetId* 和 *Encoding* 字段提供的字符集和编码，或者 MQMD 结构中的那些字段 (如果 MQWIH 位于应用程序消息数据的开头) 提供的编码。

字符集必须是对队列名称中有效的字符具有单字节字符的字符集。

用法: 如果消息将由 z/OS 工作负载管理器处理，那么该消息必须以 MQWIH 结构开头。

MQWIH 的字段

MQWIH 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

CodedCharSetId (MQLONG)

这指定遵循 MQWIH 结构的数据的字符集标识; 它不适用于 MQWIH 结构本身中的字符数据。

在 MQPUT 或 MQPUT1 调用上，应用程序必须将此字段设置为适合于数据的值。可以使用以下特殊值:

MQCCSI_INHERIT

遵循 此结构的数据中的字符数据与此结构位于同一字符集中。

队列管理器将消息中发送的结构中的此值更改为结构的实际字符集标识。如果未发生错误，那么 MQGET 调用不会返回值 MQCCSI_INHERIT。

如果 MQMD 中 *PutApplType* 字段的值为 MQAT_BROKER，那么无法使用 MQCCSI_INHERIT。
此字段的初始值为 MQCCSI_UNDEFINED。

Encoding (MQLONG)

这指定遵循 MQWIH 结构的数据的数字编码；它不适用于 MQWIH 结构本身中的数字数据。
在 MQPUT 或 MQPUT1 调用上，应用程序必须将此字段设置为适合于数据的值。
此字段的初始值为 0。

Flags (MQLONG)

该值必须为：

MQWIH_NONE

没有标志。

此字段的初始值为 MQWIH_NONE。

Format (MQCHAR8)

这将指定遵循 MQWIH 结构的数据的格式名称。

在 MQPUT 或 MQPUT1 调用上，应用程序必须将此字段设置为适合于数据的值。此字段的编码规则与 MQMD 中 *Format* 字段的编码规则相同。

此字段的长度由 MQ_FORMAT_LENGTH 指定。此字段的初始值为 MQFMT_NONE。

MsgToken (MQBYTE16)

这是唯一标识消息的消息令牌。

对于 MQPUT 和 MQPUT1 调用，将忽略此字段。此字段的长度由 MQ_MSG_TOKEN_LENGTH 指定。此字段的初始值为 MQMTOK_NONE。

保留 (MQCHAR32)

这是保留字段；必须为空白。

ServiceName (MQCHAR32)

这是要处理消息的服务的名称。

此字段的长度由 MQ_SERVICE_NAME_LENGTH 指定。此字段的初始值为 32 个空白字符。

ServiceStep (MQCHAR8)

这是与消息相关的 *ServiceName* 步骤的名称。

此字段的长度由 MQ_SERVICE_STEP_LENGTH 给出。此字段的初始值为 8 个空白字符。

StrucId (MQCHAR4)

这是结构标识。该值必须为：

MQWIH_STRUC_ID

工作信息头结构的标识。

对于 C 编程语言，还定义了常量 MQWIH_STRUC_ID_ARRAY；此值与 MQWIH_STRUC_ID 相同，但是字符数组而不是字符串。

此字段的初始值为 MQWIH_STRUC_ID。

StrucLength (MQLONG)

这是 MQWIH 结构的长度。该值必须为：

MQWIH_LENGTH_1

version-1 工作信息头结构的长度。

以下常量指定当前版本的长度:

MQWIH_CURRENT_LENGTH

当前版本的工作信息头结构的长度。

此字段的初始值为 MQWIH_LENGTH_1。

Version (MQLONG)

这是结构版本号。该值必须为:

MQWIH_VERSION_1

Version-1 工作信息头结构。

以下常量指定当前版本的版本号:

MQWIH_CURRENT_VERSION

当前版本的工作信息头结构。

此字段的初始值为 MQWIH_VERSION_1。

MQWIH 的初始值和语言声明

| 表 558: MQWIH 的 MQWIH 中字段的初始值 | | |
|------------------------------|------------------|--------|
| 字段名称 | 常量的名称 | 常量值 |
| <i>StrucId</i> | MQWIH_STRUC_ID | 'WIH~' |
| <i>Version</i> | MQWIH_VERSION_1 | 1 |
| <i>StrucLength</i> | MQWIH_LENGTH_1 | 120 |
| <i>Encoding</i> | None | 0 |
| <i>CodedCharSetId</i> | MQCCSI_UNDEFINED | 0 |
| <i>Format</i> | MQFMT_NONE | 空白 |
| <i>Flags</i> | MQWIH_NONE | 0 |
| <i>ServiceName</i> | None | 空白 |
| <i>ServiceStep</i> | None | 空白 |
| <i>MsgToken</i> | MQMTOK_NONE | Null |
| <i>Reserved</i> | None | 空白 |

注意:

1. 符号 ~ 表示单个空白字符。
2. 在 C 编程语言中, 宏变量 MQWIH_DEFAULT 包含以上列出的值。通过以下方式使用它来为结构中的字段提供初始值:

```
MQWIH MyWIH = {MQWIH_DEFAULT};
```

C 声明

```
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 声明

```

** 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-STRULENGTH 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).

```

PL/I 声明

```

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 */

```

High Level Assembler 声明

```

MQWIH          DSECT
MQWIH_STRUCID  DS CL4  Structure identifier
MQWIH_VERSION  DS F    Structure version number
MQWIH_STRULENGTH 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

```

```

MQWIH_AREA          ORG MQWIH
                    DS   CL(MQWIH_LENGTH)

```

Visual Basic 声明

```

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-出口参数块

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|----------------------|----------|------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>ExitId</i> | 出口标识 | ExitId |
| <i>ExitReason</i> | 出口调用的原因 | ExitReason |
| <i>ExitResponse</i> | 来自出口的响应 | ExitResponse |
| <i>ExitCommand</i> | API 调用代码 | ExitCommand |
| <i>ExitParmCount</i> | 参数计数 | ExitParm 计数 |
| <i>ExitUserArea</i> | 用户区域 | ExitUserArea |

MQXP 概述

可用性: z/OS。

用途:MQXP 结构用作 API 交叉输出的输入/输出参数。有关此输出的更多信息，请参阅 [API 交叉输出](#)。

字符集和编码:MQXP 中的字符数据位于本地队列管理器的字符集中;这是由 *CodedCharSetId* 队列管理器属性提供的。MQXP 中的数字数据采用本机机器编码;这是由 MQENC_NATIVE 提供的。

MQXP 的字段

MQXP 结构包含以下字段;这些字段按字母顺序进行描述:

ExitCommand (MQLONG)

此字段是在进入出口例程时设置的。它标识导致调用输出的 API 调用:

MQXC_CALLBACK

CALLBACK 调用。

MQXC_MQBACK

MQBACK 调用。

MQXC_MQCB

MQCB 调用。

MQXC_MQCLOSE

MQCLOSE 调用。

MQXC_MQCMIT

MQCMIT 调用。

MQXC_MQCTL

MQCTL 调用。

MQXC_MQGET

MQGET 调用。

MQXC_MQINQ

MQINQ 调用。

MQXC_MQOPEN

MQOPEN 调用。

MQXC_MQPUT

MQPUT 调用。

MQXC_MQPUT1

MQPUT1 调用。

MQXC_MQSET

MQSET 调用。

MQXC_MQSTAT

MQSTAT 调用。

MQXC_MQSUB

MQSUB 调用。

MQXC_MQSUBRQ

MQSUBRQ 调用。

这是出口的输出字段。

ExitId (MQLONG)

这是在进入出口例程时设置的，并指示出口类型：

MQXT_API_CROSSING_EXIT

CICS 的 API 交叉出口。

这是出口的输出字段。

ExitParm 计数 (MQLONG)

此字段是在进入出口例程时设置的。它包含 MQ 调用所采用的参数数量。这些字段为：

| 调用名称 | 参数数目 |
|---------|------|
| MQBACK | 3 |
| MQCLOSE | 5 |
| MQCMIT | 3 |
| MQGET | 9 |
| MQINQ | 10 |
| MQOPEN | 6 |
| MQPUT | 8 |
| MQPUT1 | 8 |
| MQSET | 10 |

这是出口的输入字段。

ExitReason (MQLONG)

这是在进入出口例程时设置的。对于 API 交叉出口，它指示在执行 API 调用之前还是之后调用例程：

MQXR_BEFORE

在 API 执行之前。

MQXR_AFTER

在 API 执行之后。

这是出口的输入字段。

ExitResponse (MQLONG)

该值由出口设置以与调用者通信。已定义下列值：

MQXCC_OK

出口已成功完成。

MQXCC_SUPPRESS_FUNCTION

抑制函数。

当此值由称为在 API 调用之前的 API 交叉出口设置时，不会执行 API 调用。调用的 *CompCode* 设置为 MQCC_FAILED，*Reason* 设置为 MQRC_SUPPRESSED_BY_EXIT，所有其他参数在退出时保留。

当此值由名为 *after* API 调用的 API 交叉出口设置时，队列管理器将忽略此值。

MQXCC_SKIP_FUNCTION

跳过函数。

当此值由称为在 API 调用之前的 API 交叉出口设置时，不会执行 API 调用；*CompCode* 和 *Reason* 以及所有其他参数将保留在该出口中。

当此值由名为 *after* API 调用的 API 交叉出口设置时，队列管理器将忽略此值。

这是输出的输出字段。

ExitUser 区域 (MQBYTE16)

这是可供出口使用的字段。在第一次调用任务的出口之前，该字段的长度将初始化为二进制零，此后出口对该字段所作的任何更改都将在出口的调用中保留。定义了以下值：

MQXUA_NONE

无用户信息。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQXUA_NONE_ARRAY；此值与 MQXUA_NONE 相同，但是字符数组而不是字符串。

此字段的长度由 MQ_EXIT_USER_AREA_LENGTH 指定。这是出口的输入/输出字段。

保留 (MQLONG)

这是保留字段。其值对出口不重要。

StrucId (MQCHAR4)

这是结构标识。该值必须为：

MQXP_STRUC_ID

出口参数结构的标识。

对于 C 编程语言，还定义了常量 MQXP_STRUC_ID_ARRAY；此值与 MQXP_STRUC_ID 相同，但是字符数组而不是字符串。

这是出口的输入字段。

Version (MQLONG)

这是结构版本号。该值必须为:

MQXP_VERSION_1

出口参数-块结构的版本号。

注: 当引入此结构的新版本时, 不会更改现有部件的布局。因此, 出口必须检查版本号是否等于或大于包含出口需要使用的字段的最低版本。

这是出口的输入字段。

语言声明

此结构在以下编程语言中受支持。

C 声明

```
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 声明

```
** 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).
```

PL/I 声明

```
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 */
```

High Level Assembler 声明

```

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-传输队列头

下表汇总了结构中的字段。

| 字段 | 描述 | Topic |
|-----------------------|------------|--------------------------------|
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>RemoteQName</i> | 目标队列的名称 | RemoteQName |
| <i>RemoteQMgrName</i> | 目标队列管理器的名称 | RemoteQmgrName |
| <i>MsgDesc</i> | 原始消息描述符 | MsgDesc |

MQXQH 概述

可用性: 所有 WebSphere MQ 系统和 WebSphere MQ 客户机。

用途: MQXQH 结构描述在消息位于传输队列上时，以消息的应用程序消息数据为前缀的信息。传输队列是一种特殊类型的本地队列，它临时保存发往远程队列（即，发往不属于本地队列管理器的队列）的消息。传输队列由值为 MQUS_TRANSMISSION 的 *Usage* 队列属性表示。

格式名称: MQFMT_XMIT_Q_HEADER。

字符集和编码: MQXQH 中的数据必须包含由 *CodedCharSetId* queue-manager 属性提供的字符集以及由 MQENC_NATIVE 提供的本地队列管理器的编码。

将 MQXQH 的字符集和编码设置为 *CodedCharSetId* 和 *Encoding* 字段:

- 单独的 MQMD (如果 MQXQH 结构位于消息数据的开头)，或者
- MQXQH 结构之前的头结构 (所有其他情况)。

用法: 传输队列上的消息具有两个消息描述符:

- 一个消息描述符与消息数据分开存储; 这称为单独的消息描述符，由队列管理器在将消息放入传输队列时生成。单独的消息描述符中的某些字段是从应用程序在 MQPUT 或 MQPUT1 调用上提供的消息描述符复制的。

单独的消息描述符是从传输队列中除去消息时在 MQGET 调用的 *MsgDesc* 参数中返回到应用程序的消息描述符。

- 第二个消息描述符作为消息数据的一部分存储在 MQXQH 结构中; 这称为嵌入式消息描述符，是应用程序在 MQPUT 或 MQPUT1 调用上提供的消息描述符的副本 (带有次要变体)。

嵌入式消息描述符始终是 version-1 MQMD。如果应用程序放入的消息具有 MQMD 中的一个或多个 version-2 字段的非缺省值，那么 MQMDE 结构将跟在 MQXQH 之后，并依次跟有应用程序消息数据 (如果有)。MQMDE 为:

- 由队列管理器生成 (如果应用程序使用 version-2 MQMD 来放置消息)，或者

- 在应用程序消息数据开始时已存在 (如果应用程序使用 version-1 MQMD 来放置消息)。

嵌入式消息描述符是从最终目标队列中除去消息时, 在 MQGET 调用的 *MsgDesc* 参数中返回到应用程序的消息描述符。

单独的消息描述符中的字段: 单独的消息描述符中的字段由队列管理器设置, 如下所示。如果队列管理器不支持 version-2 MQMD, 那么将在不丢失功能的情况下使用 version-1 MQMD。

| 单独 MQMD 中的字段 | 使用的值 |
|-------------------------|--|
| <i>StrucId</i> | MQMD_STRUC_ID |
| <i>Version</i> | MQMD_VERSION_2 |
| <i>Report</i> | 从嵌入式消息描述符复制, 但将 MQRO_ACCEPT_UNSUP_IF_XMIT_MASK 标识的位设置为零。(这将防止在传输队列上放置消息或从传输队列中除去消息时生成 COA 或 COD 报告消息。) |
| <i>MsgType</i> | 已从嵌入式消息描述符复制。 |
| <i>Expiry</i> | 已从嵌入式消息描述符复制。 |
| <i>Feedback</i> | 已从嵌入式消息描述符复制。 |
| <i>Encoding</i> | MQENC_NATIVE (请参阅注释) |
| <i>CodedCharSetId</i> | 队列管理器的 <i>CodedCharSetId</i> 属性。 |
| <i>Format</i> | MQFMT_XMIT_Q_HEADER |
| <i>Priority</i> | 已从嵌入式消息描述符复制。 |
| <i>Persistence</i> | 已从嵌入式消息描述符复制。 |
| <i>MsgId</i> | 队列管理器将生成新值。此消息标识与队列管理器可能为嵌入式消息描述符生成的 <i>MsgId</i> 不同 (请参阅上文)。 |
| <i>CorrelId</i> | 来自嵌入式消息描述符的 <i>MsgId</i> 。对于要放入 SYSTEM.CLUSTER.TRANSMIT.QUEUE, <i>CorrelId</i> 保留供内部使用。 |
| <i>BackoutCount</i> | 0 |
| <i>ReplyToQ</i> | 已从嵌入式消息描述符复制。 |
| <i>ReplyToQMGR</i> | 已从嵌入式消息描述符复制。 |
| <i>UserIdentifier</i> | 已从嵌入式消息描述符复制。 |
| <i>AccountingToken</i> | 已从嵌入式消息描述符复制。对于要放入 SYSTEM.CLUSTER.TRANSMIT.QUEUE, <i>AccountingToken</i> 保留供内部使用。 |
| <i>ApplIdentityData</i> | 已从嵌入式消息描述符复制。 |
| <i>PutApplType</i> | MQAT_QMGR |
| <i>PutApplName</i> | 队列管理器名称的前 28 个字节。 |
| <i>PutDate</i> | 将消息放入传输队列的日期。 |
| <i>PutTime</i> | 将消息放入传输队列的时间。 |
| <i>ApplOriginData</i> | 空白 |
| <i>GroupId</i> | MQGI_NONE |
| <i>MsgSeqNumber</i> | 1 |
| <i>Offset</i> | 0 |
| <i>MsgFlags</i> | MQMF_NONE |

| 单独 MQMD 中的字段 | 使用的值 |
|-----------------------|----------------|
| <i>OriginalLength</i> | MQOL_UNDEFINED |

- 在 Windows 上，Micro Focus COBOL 的 MQENC_NATIVE 的值与 C 的值不同。在这些环境中，单独的消息描述符中的 *Encoding* 字段中的值始终是 C 的值；此值为十进制 546。此外，MQXQH 结构中的整数字段采用与此值相对应的编码（本机 Intel 编码）。

嵌入式消息描述符中的字段: 嵌入式消息描述符中的字段与 MQPUT 或 MQPUT1 调用的 *MsgDesc* 参数中的字段具有相同的值，但以下值除外：

- Version* 字段始终具有值 MQMD_VERSION_1。
- 如果 *Priority* 字段的值为 MQPRI_PRIORITY_AS_Q_DEF，那么它将替换为队列的 *DefPriority* 属性的值。
- 如果 *Persistence* 字段具有值 MQPER_PERSISTENCE_AS_Q_DEF，那么它将替换为队列的 *DefPersistence* 属性的值。
- 如果 *MsgId* 字段具有值 MQMI_NONE，或者指定了 MQPMO_NEW_MSG_ID 选项，或者消息是分发列表消息，那么 *MsgId* 将替换为队列管理器生成的新消息标识。

当分发列表消息拆分为放置在不同传输队列上的较小分发列表消息时，每个新的嵌入式消息描述符中的 *MsgId* 字段与原始分发列表消息中的相同。

- 如果指定了 MQPMO_NEW_CORREL_ID 选项，那么 *CorrelId* 将替换为队列管理器生成的新相关标识。
- 上下文字段由 *PutMsgOpts* 参数中指定的 MQPMO_*_CONTEXT 选项指示设置；上下文字段为：

- *AccountingToken*
- *ApplIdentityData*
- *ApplOriginData*
- *PutApplName*
- *PutApplType*
- *PutDate*
- *PutTime*
- *UserIdentifier*

- 如果一个或多个 version-2 字段具有非缺省值，那么 version-2 字段（如果存在）将从 MQMD 中移除，并移至 MQMDE 结构中。

将消息放入远程队列: 当应用程序将消息放入远程队列（通过直接指定远程队列的名称或使用远程队列的本地定义）时，本地队列管理器：

- 创建包含嵌入式消息描述符的 MQXQH 结构
- 如果需要 MQMDE 并且该 MQMDE 尚不存在，那么追加该 MQMDE
- 附加应用程序消息数据
- 将消息放在相应的传输队列上

将消息直接放入传输队列: 应用程序还可以将消息直接放入传输队列。在这种情况下，应用程序必须以 MQXQH 结构作为应用程序消息数据的前缀，并使用相应的值初始化字段。此外，MQPUT 或 MQPUT1 调用的 *MsgDesc* 参数中的 *Format* 字段必须具有值 MQFMT_XMIT_Q_HEADER。

应用程序创建的 MQXQH 结构中的字符数据必须位于本地队列管理器（由 *CodedCharSetId* 队列管理器属性定义）的字符集中，并且整数数据必须采用本机机器编码。此外，MQXQH 结构中的字符数据必须用空白填充到定义的字段长度；不得使用空字符过早结束数据，因为队列管理器不会将空字符和后续字符转换为 MQXQH 结构中的空白。

但是，队列管理器不会检查是否存在 MQXQH 结构，或者是否为字段指定了有效值。

应用程序不应将其消息直接放入 SYSTEM.CLUSTER.TRANSMIT.QUEUE。

从传输队列获取消息: 从传输队列获取消息的应用程序必须以适当的方式处理 MQXQH 结构中的信息。MQXQH 结构在应用程序消息数据开头的存在由 MQGET 调用的 *MsgDesc* 参数的 *Format* 字段中返回的值

MQFMT_XMIT_Q_HEADER 指示。在 *MsgDesc* 参数的 *CodedCharSetId* 和 *Encoding* 字段中返回的值指示 MQXQH 结构中的字符和整数数据的字符集和编码。应用程序消息数据的字符集和编码由嵌入式消息描述符中的 *CodedCharSetId* 和 *Encoding* 字段定义。

MQXQH 的字段

MQXQH 结构包含以下字段; 这些字段按 **字母顺序** 进行描述:

MsgDesc (MQMD1)

这是嵌入式消息描述符, 并且是消息描述符 MQMD 的关闭副本, 该消息描述符在最初将消息放入远程队列时指定为 MQPUT 或 MQPUT1 调用上的 *MsgDesc* 参数。

注: 这是 version-1 MQMD。

此结构中字段的初始值与 MQMD 结构中的初始值相同。

RemoteQMgr 名称 (MQCHAR48)

这是拥有作为消息的明显最终目标的队列的队列管理器或队列共享组的名称。

如果消息是分发列表消息, 那么 *RemoteQMgrName* 为空白。

此字段的长度由 MQ_Q_MGR_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串, 在其他编程语言中为 48 个空白字符。

RemoteQName (MQCHAR48)

这是作为消息的明显最终目标的消息队列的名称 (例如, 如果在 *RemoteQMgrName* 将此队列定义为另一个远程队列的本地定义, 那么这可能证明不是最终目标)。

如果消息是分发列表消息 (即, 嵌入式消息描述符中的 *Format* 字段为 MQFMT_DIST_HEADER), 那么 *RemoteQName* 为空白。

此字段的长度由 MQ_Q_NAME_LENGTH 提供。此字段的初始值是 C 中的空字符串, 在其他编程语言中为 48 个空白字符。

StrucId (MQCHAR4)

这是结构标识。该值必须为:

MQXQH_STRUC_ID

传输队列头结构的标识。

对于 C 编程语言, 还定义了常量 MQXQH_STRUC_ID_ARRAY; 此值与 MQXQH_STRUC_ID 相同, 但是字符数组而不是字符串。

此字段的初始值为 MQXQH_STRUC_ID。

Version (MQLONG)

这是结构版本号。该值必须为:

MQXQH_VERSION_1

传输队列头结构的版本号。

以下常量指定当前版本的版本号:

MQXQH_CURRENT_VERSION

当前版本的传输队列头结构。

此字段的初始值为 MQXQH_VERSION_1。

MQXQH 的初始值和语言声明

表 561: MQXQH 的 MQXQH 中字段的初始值

| 字段名称 | 常量的名称 | 常量值 |
|----------------|--|---------|
| StrucId | MQXQH_STRUC_ID | 'XQH~' |
| Version | MQXQH_VERSION_1 | 1 |
| RemoteQName | None | 空字符串或空白 |
| RemoteQMgrName | None | 空字符串或空白 |
| MsgDesc | 名称和值与 MQMD 相同; 请参阅 第 402 页的表 515 | - |

注意:

1. 符号 ~ 表示单个空白字符。
2. 值 Null 字符串或空白表示 C 中的空字符串, 而空白字符表示其他编程语言中的空字符。
3. 在 C 编程语言中, 宏变量 MQXQH_DEFAULT 包含以上列出的值。 通过以下方式使用它来为结构中的字段提供初始值:

```
MQXQH MyXQH = {MQXQH_DEFAULT};
```

C 声明

```
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 声明

```
** 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).

```

PL/I 声明

```

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,
  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 */

```

High Level Assembler 声明

```

MQXQH          DSECT
MQXQH_STRUCID  DS   CL4   Structure identifier
MQXQH_VERSION  DS   F     Structure version number
MQXQH_REMOTENAME 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) | |

Visual Basic 声明

```

Type MQXQH
  StructId      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

```

函数调用

本部分提供了有关所有可能的 MQI 调用的信息。针对每个不同的调用提供了每种可能语言的描述，语法，参数信息，用法说明和语言调用。

呼叫描述

本节描述 MQI 调用。

- [第 548 页的『MQBACK-回退更改』](#)
- [第 552 页的『MQBEGIN-开始工作单元』](#)
- [第 555 页的『MQBUFMH - 将缓冲区转换为消息句柄』](#)
- [第 558 页的『MQCB-管理回调』](#)
- [第 567 页的『MQCB_FUNCTION-回调函数』](#)
- [第 568 页的『MQCLOSE-关闭对象』](#)
- [第 575 页的『MQCMIT-落实更改』](#)
- [第 578 页的『MQCONN - 连接队列管理器』](#)

- [第 585 页的『MQCONN-连接队列管理器 \(扩展\)』](#)
- [第 591 页的『MQCRTMH-创建消息句柄』](#)
- [第 594 页的『MQCTL-控制回调』](#)
- [第 599 页的『MQDISC-断开连接队列管理器』](#)
- [第 603 页的『MQDLTMH-删除消息句柄』](#)
- [第 605 页的『MQDLTMP-删除消息属性』](#)
- [第 607 页的『MQGET-获取消息』](#)
- [第 619 页的『MQINQ-查询对象属性』](#)
- [第 633 页的『MQINQMP-查询消息属性』](#)
- [第 638 页的『MQMHBUF-将消息句柄转换为缓冲区』](#)
- [第 642 页的『MQOPEN-打开对象』](#)
- [第 658 页的『MQPUT-放置消息』](#)
- [第 670 页的『MQPUT1 -放置一条消息』](#)
- [第 679 页的『MQSET-设置对象属性』](#)
- [第 685 页的『MQSETMP-设置消息属性』](#)
- [第 689 页的『MQSTAT-检索状态信息』](#)
- [第 638 页的『MQMHBUF-将消息句柄转换为缓冲区』](#)
- [第 693 页的『MQSUB-注册预订』](#)
- [第 698 页的『MQSUBRQ-预订请求』](#)

UNIX 平台上以 *man* 页面形式提供的联机帮助可用于这些调用。

注: 与数据转换, MQXCNCV 和 MQ_DATA_CONV_EXIT 关联的调用位于 [第 791 页的『数据转换出口』](#) 中。

调用描述中使用的约定

对于每个调用, 此主题集合以独立于编程语言的格式提供调用的参数和用法的描述。其次是在每种受支持的编程语言中调用的典型调用及其参数的典型声明。

要点: 对 WebSphere MQ API 调用进行编码时, 必须确保提供所有相关参数 (如以下部分中所述)。如果无法执行此操作, 那么可能会产生不可预测的结果。

每个调用的描述都包含以下部分:

调用名称

调用名称, 后跟对调用目的的简要描述。

参数

对于每个参数, 名称后跟括号 () 中的数据类型 以及下列其中一项:

输入

在进行调用时, 在参数中提供信息。

输出

当调用完成或失败时, 队列管理器将返回参数中的信息。

输入/输出

您在进行调用时在参数中提供信息, 当调用完成或失败时, 队列管理器会更改信息。

例如:

Compcode (MQLONG)-输出

在某些情况下, 数据类型是结构。在所有情况下, 都有关于 [第 214 页的『基本数据类型』](#) 中的数据类型或结构的更多信息。

每个调用中的最后两个参数是完成代码和原因码。完成代码指示调用是否已成功完成，部分完成或根本未完成。在原因码中提供了有关调用的部分成功或失败的更多信息。有关每个完成代码和原因码的更多信息，请参阅第 764 页的『返回码』。

使用说明

有关调用的其他信息，描述如何使用该调用以及对其使用的任何限制。

汇编程序语言调用

调用的典型调用及其参数的声明 (使用汇编语言)。

C 调用

调用的典型调用及其参数的声明 (以 C 为代表)。

COBOL 调用

在 COBOL 中调用调用及其参数的典型调用。

PL/I 调用

调用的典型调用及其参数的声明，以 PL/I 表示。

所有参数都通过引用传递。

Visual Basic 调用

在 Visual Basic 中对调用的典型调用及其参数的声明。

其他表示法约定包括:

常量

常量的名称以大写形式显示; 例如, MQOO_OUTPUT。具有相同前缀的一组常量如下所示: MQIA_*。请参阅第 49 页的『常量』以获取常量的值。

数组

在某些调用中, 参数是不具有固定大小的字符串数组。在这些参数的描述中, 小写 n 表示数字常量。对该参数的声明进行编码时, 请将 n 替换为所需的数字值。

使用 C 语言中的调用

仅输入且类型为 MQHCONN, MQHOBJ, MQHMSG 或 MQLONG 的参数按值传递。对于所有其他参数, 将按值传递参数的地址。

您不需要指定每次调用函数时按地址传递的所有参数。在不需要特定参数的情况下, 指定空指针作为函数调用上的参数, 以代替参数数据的地址。可以进行此操作的参数在调用描述中被识别。

不会返回任何参数作为调用的值; 在 C 术语中, 这意味着所有调用都返回 void。

声明 Buffer 参数

MQGET, MQPUT 和 MQPUT1 调用各有一个具有未定义数据类型的参数: Buffer 参数。使用此参数可发送和接收应用程序的消息数据。

此类参数在 C 示例中显示为 MQBYTE 的数组。您可以通过这种方式声明参数, 但通常更便于将它们声明为描述消息中数据布局的特定结构。函数原型将参数声明为指向 void 的指针, 以便您可以将任何类型的数据的地址指定为调用上的参数。

"指针到空" 是指向未定义格式的数据的指针。定义为:

```
typedef void *PMQVOID;
```

MQBACK-回退更改

MQBACK 调用向队列管理器指示将回退自最后一个同步点以来发生的所有消息获取和放置。

将删除作为工作单元的一部分放入的消息; 将在队列中恢复作为工作单元的一部分检索的消息。

- 在 z/OS 上, 此调用仅由批处理程序 (包括 IMS 批处理 DL/I 程序) 使用。
- 在 IBM i 上, 以兼容性方式运行的应用程序不支持此调用。

语法

MQBACK (*Hconn*, *Compcode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNEX 调用返回了 *Hconn* 的值。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') 调用在环境中无效。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_OBJECT_DAMAGED

(2101, X'835') 对象已损坏。

MQRC_OUTCOME_MIXED

(2123, X'84B') 落实或回退操作的结果是混合的。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部存储介质已满。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅 [原因码](#)

使用说明

1. 仅当队列管理器自身协调工作单元时，才能使用此调用。这可以是：

- 本地工作单元，其中更改仅影响 MQ 资源。
- 全局工作单元，其中的更改可能会影响属于其他资源管理器的资源以及影响 MQ 资源。

有关本地和全局工作单元的更多详细信息，请参阅第 552 页的『MQBEGIN-开始工作单元』。

2. 在队列管理器未协调工作单元的环境中，请使用相应的回退调用而不是 MQBACK。环境还可能支持由应用程序异常终止导致的隐式回退。

- 在 z/OS 上，使用以下调用：
 - 如果工作单元仅影响 MQ 资源，那么批处理程序 (包括 IMS 批处理 DL/I 程序) 可以使用 MQBACK 调用。但是，如果工作单元同时影响 MQ 资源和属于其他资源管理器的资源 (例如，DB2)，请使用 z/OS 可恢复资源服务 (RRS) 提供的 SRRBACK 调用。SRRBACK 调用会回退对属于已启用 RRS 协调的资源管理器的资源的更改。
 - CICS 应用程序必须使用 EXEC CICS SYNCPOINT ROLLBACK 命令来回退工作单元。请勿将 MQBACK 调用用于 CICS 应用程序。
 - IMS 应用程序 (批处理 DL/I 程序除外) 必须使用 IMS 调用 (例如 ROLB) 来回退工作单元。请勿将 MQBACK 调用用于 IMS 应用程序 (批处理 DL/I 程序除外)。
- 在 IBM i 上，将此调用用于由队列管理器协调的本地工作单元。这意味着在作业级别不得存在落实定义，即，不得对作业发出带有 CMTSCOPE(*JOB) 参数的 STRCMTCTL 命令。

3. 如果应用程序在工作单元中以未落实的更改结束，那么这些更改的处置取决于应用程序是正常结束还是异常结束。请参阅第 599 页的『MQDISC-断开连接队列管理器』中的用法说明以获取更多详细信息。

4. 当应用程序在逻辑消息的组或段中放置或获取消息时，队列管理器会保留与消息组和上次成功 MQPUT 和 MQGET 调用的逻辑消息相关的信息。此信息与队列句柄相关联，并包括如下内容：

- MQMD 中 *GroupId*、*MsgSeqNumber*、*Offset* 和 *MsgFlags* 字段的值。
- 消息是否是工作单元的一部分。
- 对于 MQPUT 调用：消息是持久消息还是非持久消息。

队列管理器保留三组组和段信息，其中一组针对以下各项：

- 最后一次成功的 MQPUT 调用 (这可以是工作单元的一部分)。
- 从队列中除去消息的最后一次成功 MQGET 调用 (这可以是工作单元的一部分)。
- 上次成功的 MQGET 调用浏览了队列上的消息 (此不能是工作单元的一部分)。

5. 与 MQGET 调用相关联的信息将复原为当前工作单元中该队列句柄的第一次成功 MQGET 调用之前的值。

在工作单元启动后由应用程序更新但在工作单元作用域之外的队列，如果工作单元回退，那么不会恢复其组和段信息。

当回退工作单元时，将组和段信息恢复到其先前的值允许应用程序在多个工作单元之间传播由多个段组成的大型消息组或大型逻辑消息，并在其中一个工作单元发生故障时在消息组或逻辑消息中的正确位置重新启动。

如果本地队列管理器只有有限的队列存储器，那么使用多个工作单元可能是有利的。但是，应用程序必须保留足够的信息，以便在发生系统故障时能够在正确的位置重新启动放入或获取消息。

有关如何在系统故障后的正确位置重新启动的详细信息，请参阅第 433 页的『MQPMO-Put-message 选项』中描述的 MQPMO_LOGICAL_ORDER 选项和第 324 页的『MQGMO-Get-消息选项』中描述的 MQGMO_LOGICAL_ORDER 选项。

仅当队列管理器协调工作单元时，其余使用说明才适用。

6. 工作单元具有与连接句柄相同的作用域。必须使用同一连接句柄来执行影响特定工作单元的所有 MQ 调用。使用另一个连接句柄发出的调用 (例如，另一个应用程序发出的调用) 会影响另一个工作单元。请参阅第 578 页的『MQCONN - 连接队列管理器』中描述的 *Hconn* 参数，以获取有关连接句柄作用域的信息。
7. 只有作为当前工作单元的一部分放入或检索的消息才会受此调用影响。
8. 在工作单元中发出 MQGET，MQPUT 或 MQPUT1 调用但从未发出落实或回退调用的长时间运行的应用程序可以使用不可用于其他应用程序的消息填充队列。为了避免这种可能性，管理员必须将 *MaxUncommittedMsgs queue-manager* 属性设置为足以防止失控应用程序填充队列的值，但设置为足以允许期望的消息传递应用程序正常工作的值。

C 调用

```
MQBACK (Hconn, &CompCode, &Reason);
```

按如下所示声明参数:

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

COBOL 调用

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.
```

PL/I 调用

```
call MQBACK (Hconn, CompCode, Reason);
```

按如下所示声明参数:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

高级汇编程序调用

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

按如下所示声明参数:

```
HCONN      DS F Connection handle
COMPCODE   DS F Completion code
REASON     DS F Reason code qualifying COMPCODE
```

Visual Basic 调用

```
MQBACK Hconn, CompCode, Reason
```

按如下所示声明参数:

```
Dim Hconn As Long 'Connection handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQBEGIN-开始工作单元

MQBEGIN 调用开始一个由队列管理器协调的工作单元, 该工作单元可能涉及外部资源管理器。

语法

```
MQBEGIN (Hconn, BeginOptions, Compcode, Reason)
```

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。

Hconn 必须是非共享连接句柄。如果指定了共享连接句柄, 那么调用将失败, 原因码为 MQRC_HCONN_ERROR。请参阅第 283 页的『MQCNO - 连接选项』中 MQCNO_HANDLE_SHARE_* 选项的描述, 以获取有关共享和非共享句柄的更多信息。

BeginOptions

类型:MQBO-输入/输出

这些选项用于控制 MQBEGIN 的操作, 如第 248 页的『MQBO-开始选项』中所述。

如果不需要任何选项, 那么以 C 或 S/390 汇编程序编写的程序可以指定空参数地址, 而不是指定 MQBO 结构的地址。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_NO_EXTERNAL_参与者

(2121, X'849') 未注册参与资源管理器。

MQRC_PARTICIPANT_NOT_AVAILABLE

(2122, X'84A') 参与资源管理器不可用。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_BO_ERROR

(2134, X'856 ') 开始选项结构无效。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') 调用在环境中无效。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_OPTIONS_ERROR

(2046, X'7FE') 选项无效或不一致。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

MQRC_UOW_IN_PROGRESS

(2128, X'850 ') 工作单元已开始。

有关这些原因码的更多信息, 请参阅 [原因码](#)。

使用说明

1. 使用 MQBEGIN 调用来启动由队列管理器协调的工作单元, 该工作单元可能涉及对其他资源管理器拥有的资源的更改。队列管理器支持三种类型的工作单元:
 - **队列管理器协调的本地工作单元:** 一个工作单元, 其中队列管理器是唯一参与的资源管理器, 因此队列管理器充当工作单元协调程序。
 - 要启动此类型的工作单元, 请在工作单元中的第一个 MQPUT, MQPUT1 或 MQGET 调用上指定 MQPMO_SYNCPOINT 或 MQGMO_SYNCPOINT 选项。
 - 要落实或回退此类型的工作单元, 请使用 MQCMIT 或 MQBACK 调用。
 - **队列管理器协调的全局工作单元:** 队列管理器充当 MQ 资源和属于其他资源管理器的资源的工作单元协调程序的工作单元。这些资源管理器与队列管理器协作, 以确保对工作单元中资源的所有更改都已落实或一起回退。
 - 要启动此类型的工作单元, 请使用 MQBEGIN 调用。
 - 要落实或回退此类型的工作单元, 请使用 MQCMIT 和 MQBACK 调用。
 - **外部协调的全局工作单元:** 一个工作单元, 其中队列管理器是参与者, 但队列管理器不充当工作单元协调程序。而是有一个外部工作单元协调程序, 队列管理器与该协调程序协作。
 - 要启动此类型的工作单元, 请使用外部工作单元协调程序提供的相关调用。
- 如果使用 MQBEGIN 调用来尝试启动工作单元, 那么调用将失败, 原因码为 MQRC_environment_ERROR。

- 要落实或回退此类型的工作单元，请使用外部工作单元协调程序提供的落实和回退调用。
如果使用 MQCMIT 或 MQBACK 调用来落实或回退工作单元，那么调用将失败，原因码为 MQRC_ENVIRONMENT_ERROR。
- 2. 如果应用程序在工作单元中以未落实的更改结束，那么这些更改的处置取决于应用程序是正常结束还是异常结束。请参阅第 599 页的『MQDISC-断开连接队列管理器』中的用法说明以获取更多详细信息。
- 3. 一个应用程序一次只能参与一个工作单元。MQBEGIN 调用失败，原因码为 MQRC_UOW_IN_PROGRESS (如果应用程序已存在工作单元)，而不管它是哪种类型的工作单元。
- 4. MQBEGIN 调用在 MQ MQI 客户机环境中无效。尝试使用调用失败，原因码为 MQRC_ENVIRONMENT_ERROR。
- 5. 当队列管理器充当全局工作单元的工作单元协调程序时，可以参与工作单元的资源管理器在队列管理器配置文件中定义。
- 6. 在 IBM i 上，支持以下三种类型的工作单元：
 - 仅当作业级别不存在落实定义时，才能使用 **队列管理器协调的本地工作单元**，即，不得对该作业发出带有 CMTSCOPE(*JOB) 参数的 STRCMTCTL 命令。
 - 不支持 **队列管理器协调的全局工作单元**。
 - **外部协调的全局工作单元** 只能在作业级别存在落实定义时使用，即必须已针对作业发出带有 CMTSCOPE(*JOB) 参数的 STRCMTCTL 命令。如果已执行此操作，那么 IBM i COMMIT 和 ROLLBACK 操作将应用于 MQ 资源以及属于其他参与资源管理器的资源。

C 调用

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

按如下所示声明参数：

```
MQHCNN  Hconn;          /* Connection handle */
MQBO    BeginOptions;  /* Options that control the action of MQBEGIN */
MQLONG  CompCode;      /* Completion code */
MQLONG  Reason;        /* Reason code qualifying CompCode */
```

COBOL 调用

```
CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

按如下所示声明参数：

```
** 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.
```

PL/I 调用

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

按如下所示声明参数：

```
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 */

```

Visual Basic 调用

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

按如下所示声明参数:

```

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 - 将缓冲区转换为消息句柄

MQBUFMH 函数调用将缓冲区转换为消息句柄，并且是 MQMHBUF 调用的逆函数。

此调用在缓冲区中获取消息描述符和 MQRFH2 属性，并通过消息句柄提供这些属性。可以选择除去消息数据中的 MQRFH2 属性。必要时，将更新消息描述符的 *Encoding*、*CodedCharSetId* 和 *Format* 字段，以在除去属性后正确描述缓冲区的内容。

语法

```
MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, Buffer, BufferLength, DataLength,
Compcode, Reason)
```

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。Hconn 的值必须与用于创建 Hmsg 参数中指定的消息句柄的连接句柄相匹配。

如果消息句柄是使用 MQHC_UNASSOCIATED_HCONN 创建的，那么必须在将缓冲区转换为消息句柄的线程上建立有效连接。如果未建立有效连接，那么调用将失败，并返回 MQRC_CONNECTION_BROKEN。

Hmsg

类型:MQHMQSG-输入

这是需要缓冲区的消息句柄。该值由先前的 MQCRTMH 调用返回。

BufMsgHOpts

类型:MQBMHO-输入

MQBMHO 结构允许应用程序指定用于控制如何从缓冲区生成消息句柄的选项。

请参阅第 246 页的『MQBMHO-缓冲区到消息句柄选项』，以了解详细信息。

MsgDesc

类型:MQMD-输入/输出

MsgDesc 结构包含消息描述符属性并描述缓冲区的内容。

在调用的输出上，可选择从缓冲区中除去属性，在这种情况下，将更新消息描述符以正确描述缓冲区。

此结构中的数据必须采用应用程序的字符集和编码。

BufferLength

类型:MQLONG-输入

BufferLength 是缓冲区的长度 (以字节为单位)。

零字节的 *BufferLength* 有效, 指示缓冲区不包含任何数据。

buffer

类型: MQBYTEExBuffer 长度-输入/输出

这些选项用于控制 MQBEGIN 的操作, 如第 552 页的『MQBEGIN-开始工作单元』中所述。

Buffer 定义包含消息缓冲区的区域。对于大多数数据, 您应该在 4 字节边界上对齐缓冲区。

如果 *Buffer* 包含字符或数字数据, 请将 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段设置为适合于数据的值; 这将允许在必要时转换数据。

如果在消息缓冲区中找到属性, 那么可以选择除去这些属性; 这些属性稍后将在从调用返回时从消息句柄变为可用。

在 C 编程语言中, 该参数被声明为指向 void 的指针, 这意味着任何类型的数据的地址都可以被指定为参数。

如果 *BufferLength* 参数为零, 那么不会引用 *Buffer*; 在此情况下, 以 C 或 System/390 汇编程序编写的程序传递的参数地址可以为空。

DataLength

类型: MQLONG - 输出

可能已除去属性的缓冲区的长度 (以字节计)。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 是 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BMHO_ERROR

(2489, X'09B9') 缓冲区到消息句柄选项结构无效。

MQRC_BUFFER_ERROR

(2004, X'07D4') 缓冲区参数无效。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') 与队列管理器的连接丢失。

MQRC_HMSG_ERROR

(2460, X'099C') 消息句柄无效。

MQRC_MD_ERROR

(2026, X'07EA') 消息描述符无效。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') 消息句柄已在使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') 选项无效或不一致。

MQRC_RFH_ERROR

(2334, X'091E') MQRFH2 结构无效。

MQRC_RFH_FORMAT_ERROR

(2421, X'0975 ') 无法解析包含属性的 MQRFH2 文件夹。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

MQBUFMH 调用无法被 API 出口拦截-缓冲区将转换为应用程序空间中的消息句柄; 该调用无法到达队列管理器。

C 调用

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,
                   BUFFER, DATALENGTH, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.

```

PL/I 调用

```

call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
             DataLength, CompCode, Reason);

```

按如下所示声明参数:

```

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 */

```

高级汇编程序调用

```

CALL MQBUFMH, (HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
             DATALENGTH,COMPCODE,REASON)

```

按如下所示声明参数:

| | | | |
|--------------|----------|-------|--|
| 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-管理回调

MQCB 调用为指定的对象句柄注册回调，并控制对该回调的激活和更改。

回调是 IBM WebSphere MQ 在发生特定事件时调用的代码段 (指定为可动态链接的函数的名称或作为函数指针)。

要在 V7 客户机上使用 MQCB 和 MQCTL，必须连接到 V7 服务器，并且通道的 **SHARECNV** 参数必须具有非零值。

可以定义的回调类型为:

消息使用者

当满足指定选择条件的消息在对象句柄上可用时，将调用消息使用者回调函数。

只能针对每个对象句柄注册一个回调函数。如果要使用多个选择标准读取单个队列，那么必须多次打开该队列并在每个句柄上注册使用者函数。

事件处理程序

将针对影响整个回调环境的条件调用事件处理程序。

当发生事件条件 (例如, 队列管理器或连接停止或停顿) 时, 将调用该函数。

对于特定于单个消息使用者的条件 (例如 MQRC_GET_INHIBITED), 不会调用该函数; 但是, 如果回调函数未正常结束, 那么会调用该函数。

语法

MQCB (*Hconn*, 操作, *CallbackDesc*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *CompCode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNEX 调用返回了 *Hconn* 的值。

在 z/OS for CICS 应用程序和 IBM i 上, 对于以兼容性方式运行的应用程序, 您可以为 *MQHC_DEF_HCONN* 指定以下特殊值以使用与此执行单元相关联的连接句柄。

operation

类型:MQLONG-输入

在为指定对象句柄定义的回调上正在处理的操作。必须指定下列其中一个选项; 如果需要多个选项, 那么值可以是:

- 相加 (不要将同一常量相加多次), 或者
- 通过按位“或”运算组合在一起 (如果编程语言支持位运算)。

MQOP_REGISTER

定义指定对象句柄的回调函数。此操作定义要调用的函数以及要使用的选择标准。

如果已经为对象句柄定义了回调函数, 那么将替换该定义。如果在替换回调时检测到错误, 那么将注销该函数。

如果在先前已注销的回调函数中注册回调, 那么会将其视为替换操作; 不会调用任何初始或最终调用。

您可以将 MQOP_REGISTER 与 MQOP_SUSPEND 或 MQOP_RESUME 配合使用。

MQOP_DEREGISTER

停止使用对象句柄的消息, 并从符合回调条件的对象句柄中除去该句柄。

如果关闭了关联的句柄, 那么将自动注销回调。

如果从使用者内部调用 MQOP_DEREGISTER, 并且回调定义了停止调用, 那么将在从使用者返回时调用该回调。

如果对没有注册使用者的 *Hobj* 发出此操作, 那么调用将返回 MQRC_CALLBACK_NOT_REGISTERED。

MQOP_SUSPEND

暂挂使用对象句柄的消息。

如果此操作应用于事件处理程序, 那么事件处理程序在暂挂时不会获取事件, 并且在恢复操作时不会向操作提供处于暂挂状态时丢失的任何事件。

暂挂时, 使用者函数将继续获取控制类型回调。

MQOP_RESUME

继续使用对象句柄的消息。

如果此操作应用于事件处理程序, 那么事件处理程序在暂挂时不会获取事件, 并且在恢复操作时不会向操作提供处于暂挂状态时丢失的任何事件。

CallbackDesc

类型:MQCBD-输入

这是一种结构, 用于标识应用程序正在注册的回调函数以及注册该回调函数时使用的选项。

请参阅 [MQCBD](#) 以获取结构的详细信息。

只有 MQOP_REGISTER 选项需要回调描述符; 如果不需要该描述符, 那么传递的参数地址可以为空。

Hobj

类型:MQHOBJ-输入

此句柄表示对要从中使用消息的对象建立的访问权。这是从先前的 [MQOPEN](#) 或 [MQSUB](#) 调用 (在 *Hobj* 参数中) 返回的句柄。

定义事件处理程序例程 (MQCBT_EVENT_HANDLER) 时不需要 *Hobj*, 应该将其指定为 MQHO_NONE。

如果已从 MQOPEN 调用返回 *Hobj*, 那么必须已使用以下一个或多个选项打开队列:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

类型:MQMD-输入

此结构描述所需消息的属性以及检索的消息的属性。

MsgDesc 参数定义使用者所需的消息属性, 以及要传递给消息使用者的 MQMD 版本。

MQMD 中的 *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber* 和 *Offset* 用于消息选择, 具体取决于 *GetMsgOpts* 参数中指定的选项。

如果指定 MQGMO_CONVERT 选项, 那么 *Encoding* 和 *CodedCharSetId* 将用于消息转换。

请参阅 [MQMD](#) 以获取详细信息。

MsgDesc 用于 MQOP_REGISTER, 如果您需要任何字段的缺省值以外的值。 *MsgDesc* 不用于事件处理程序。

如果不需要描述符, 那么传递的参数地址可以为空。

请注意, 如果针对具有重叠选择器的同一队列注册了多个使用者, 那么未定义每条消息的所选使用者。

GetMsg 选项

类型:MQGMO-输入

GetMsgOpts 参数控制消息使用者获取消息的方式。在 MQGET 调用上使用, 此参数的所有选项都具有第 324 页的『[MQGMO-Get-消息选项](#)』中描述的含义, 但以下选项除外:

MQGMO_SET_SIGNAL

不允许此选项。

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT 和 MQGMO_MARK_*

传递到浏览使用者的消息的顺序由这些选项的组合决定。重要组合包括:

MQGMO_BROWSE_FIRST

队列上的第一条消息将重复传递给使用者。当使用者以破坏性方式使用回调中的消息时, 这很有用。请谨慎使用此选项。

MQGMO_BROWSE_NEXT

将向使用者提供队列上的每条消息, 从当前光标位置直到到达队列末尾。

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT

将光标重置为队列的开始。然后, 将为使用者提供每条消息, 直到光标到达队列的末尾为止。

MQGMO_BROWSE_FIRST + MQGMO_MARK_*

从队列开始, 将为使用者提供队列上的第一条未标记的消息, 然后对此使用者进行标记。此组合确保使用者可以接收在当前光标点后面添加的新消息。

MQGMO_BROWSE_NEXT + MQGMO_MARK_*

从光标位置开始, 将为使用者提供队列上的下一条未标记的消息, 然后对此使用者进行标记。请谨慎使用此组合, 因为可以将消息添加到当前光标位置后的队列中。

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_*

不允许此组合。如果已使用，那么调用将返回 MQRC_OPTIONS_ERROR。

MQGMO_NO_WAIT, MQGMO_WAIT 和 WaitInterval

这些选项控制如何调用使用者。

MQGMO_NO_WAIT

从不使用 MQRC_NO_MSG_AVAILABLE 调用使用者。仅针对消息和事件调用使用者。

具有零 WaitInterval 的 MQGMO_WAIT

MQRC_NO_MSG_AVAILABLE 代码在没有可用消息时传递给使用者，并且自上次 "无消息" 原因码以来，使用者已启动或至少传递了一条消息。

当指定零等待时间间隔时，这将阻止使用者在繁忙循环中进行轮询。

MQGMO_WAIT 和正 WaitInterval

将在指定的等待时间间隔之后调用使用者，原因码为 MQRC_NO_MSG_AVAILABLE。无论是否已将任何消息传递给使用者，都将进行此调用。这允许用户执行脉动信号或批处理类型处理。

MQGMO_WAIT 和 MQWI_UNLIMITED 的 WaitInterval

这指定返回 MQRC_NO_MSG_AVAILABLE 之前的无限等待。从不使用 MQRC_NO_MSG_AVAILABLE 调用使用者。

GetMsgOpts 仅用于 MQOP_REGISTER，如果您需要任何字段的缺省值以外的值。*GetMsgOpts* 不用于事件处理程序。

如果不需要 *GetMsgOpts*，那么传递的参数地址可以为空。使用此参数与将 MQGMO_DEFAULT 与 MQGMO_FAIL_IF QUIESCING 一起指定相同。

如果在 MQGMO 结构中提供了消息属性句柄，那么将在传递到使用者回调的 MQGMO 结构中提供副本。从 MQCB 调用返回时，应用程序可以删除消息属性句柄。

CompCode

类型：MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_WARNING

警告（部分完成）。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

以下列表中的原因码是队列管理器可以针对 *Reason* 参数返回的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 是 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') 无法装入数据转换服务模块。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CALLBACK_LINK_ERROR

(2487, X'9B7') 回调类型字段不正确。

MQRC_CALLBACK_NOT_REGISTERED

(2448, X'990') 无法注销, 暂挂或恢复, 因为没有已注册的回调。

MQRC_CALLBACK_ROUTINE_ERROR

(2486, X'9B6') 必须指定 *CallbackFunction* 或 *CallbackName*, 但不能同时指定两者。

MQRC_CALLBACK_TYPE_ERROR

(2483, X'9B3') 回调类型字段不正确。

MQRC_CBD_OPTIONS_ERROR

(2484, X'9B4') MQCBD 选项字段不正确。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 未授权连接。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_CORREL_ID_ERROR

(2207, X'89F') 相关标识错误。

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') 数据长度参数无效。

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') 请求的功能在当前环境中不可用。

MQRC_GET_禁止

(2016, X'7E0') 队列禁止获取。

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') 全局工作单元冲突。

MQRC_GMO_ERROR

(2186, X'88A') Get-message 选项结构无效。

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') 用于全局工作单元的句柄。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_HOBJ_ERROR

(2019, X'7E3') 对象句柄无效。

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') 浏览规范不一致。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 工作单元规范不一致。

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') 游标下的消息对于检索无效。

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') 全局工作单元与本地工作单元冲突。

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') 匹配选项无效。

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B4') *MaxMsgLength* 字段不正确。

MQRC_MD_ERROR
(2026, X'7EA') 消息描述符无效。

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') 在模块中找不到指定的函数入口点。

MQRC_MODULE_INVALID
(2496, X'9C0') 找到模块, 但是它的类型错误; 不是 32 位, 64 位或有效的动态链接库。

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') 在搜索路径中找不到模块或无权装入。

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') 消息序号无效。

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') 使用消息令牌无效。

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') 无消息可用。

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') 浏览光标未定位在消息上。

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') 未打开队列以进行浏览。

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') 未打开队列以进行输入。

MQRC_OBJECT_CHANGED
(2041, X'7F9') 对象定义自打开以来已更改。

MQRC_OBJECT_DAMAGED
(2101, X'835') 对象已损坏。

MQRC_OPERATION_ERROR
(2206, X'89E') API 调用上的操作码不正确。

MQRC_OPTIONS_ERROR
(2046, X'7FE') 选项无效或不一致。

MQRC_PAGESET_ERROR
(2193, X'891') 访问页集数据集时出错。

MQRC_Q_DELETED
(2052, X'804') 队列已删除。

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') 队列具有错误的索引类型。

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING
(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR_STOPPING
(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM
(2102, X'836') 没有足够系统资源可用。

MQRC_SIGNAL_众数

(2069, X'815 ') 此句柄的信号未完成。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口程序禁止调用。

MQRC_SYNCPOINT_LIMIT_已达到

(2024, X'7E8') 无法在当前工作单元中处理更多消息。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818 ') 同步点支持不可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X' 932 ') 在全局工作单元中登记失败。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X' 933 ') 不支持混合工作单元调用。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 工作单元不可供队列管理器使用。

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') MQGMO 中的等待时间间隔无效。

MQRC_不法_gmo_version

(2256, X'8D0') 提供的 MQGMO 版本不正确。

MQRC_不法_md_version

(2257, X'8D1') 提供的 MQMD 版本不正确。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

1. MQCB 用于定义要针对每条消息调用的操作，这些操作与队列上可用的指定条件相匹配。处理操作时，将从队列中除去消息并将其传递到定义的消息使用者，或者提供用于检索消息的消息令牌。
2. MQCB 可用于在开始使用 MQCTL 之前定义回调例程，也可以从回调例程中使用 MQCB。
3. 要从回调例程外部使用 MQCB，必须首先使用 MQCTL 暂挂消息使用，然后恢复使用。
4. IMS 适配器中不支持 MQCB。

消息使用者回调序列

您可以将使用者配置为在使用者生命周期内的关键点调用回调。例如：

- 消费者首次注册时，
- 当连接启动时，
- 当连接停止时，
- 当使用者被显式注销或由 MQCLOSE 隐式注销时。

| 动词 | 含义 |
|---------------|---------------------------------|
| MQCTL (START) | 使用 MQOP_START 操作的 MQCTL 调用 |
| MQCTL (STOP) | 使用 MQOP_STOP 操作的 MQCTL 调用 |
| MQCTL (等待) | 使用 MQOP_START_WAIT 操作的 MQCTL 调用 |

这允许使用者维护与该使用者相关联的状态。当应用程序请求回调时，使用者调用的规则如下所示：

注册

始终是回调的第一种调用类型。

始终在与 MQCB (REGISTER) 调用相同的线程上调用。

START

始终与 MQCTL (START) 动词同步调用。

- 在 MQCTL (START) 动词返回之前完成所有 START 回调。

与消息传递位于同一线程上 (如果请求了 THREAD_AFFINITY)。

例如, 如果先前回调在 MQCTL (START) 期间发出 MQCTL (STOP), 那么不保证具有启动的调用。

结束

在此调用之后, 将不会再传递任何消息或事件, 直到连接重新启动为止。

如果先前针对 START, 消息或事件调用了应用程序, 那么将保证 STOP。

DEREGISTER

始终是回调的最后一种调用类型。

确保应用程序在 START 和 STOP 回调中执行基于线程的初始化和清除。可以使用 REGISTER 和 DEREGISTER 回调执行基于非线程的初始化和清除。

请勿对线程的寿命和可用性进行任何非声明的假设。例如, 不要依赖在上次调用 DEREGISTER 之后保持活动状态的线程。同样, 当您选择不使用 THREAD_AFFINITY 时, 请勿假定每当启动连接时都存在该线程。

如果应用程序对线程特征有特殊要求, 那么它可以始终相应地创建线程, 然后使用 MQCTL (WAIT)。这具有向 IBM WebSphere MQ "提供" 线程以进行异步消息传递的效果。

消息使用者连接使用情况

您可以将使用者配置为在使用者生命周期内的关键点调用回调。例如:

- 消费者首次注册时,
- 当连接启动时,
- 当连接停止时,
- 当使用者被显式注销或由 MQCLOSE 隐式注销时。

| 动词 | 含义 |
|---------------|---------------------------------|
| MQCTL (START) | 使用 MQOP_START 操作的 MQCTL 调用 |
| MQCTL (STOP) | 使用 MQOP_STOP 操作的 MQCTL 调用 |
| MQCTL (等待) | 使用 MQOP_START_WAIT 操作的 MQCTL 调用 |

这允许使用者维护与该使用者相关联的状态。当应用程序请求回调时, 使用者调用的规则如下所示:

注册

始终是回调的第一种调用类型。

始终在与 MQCB (REGISTER) 调用相同的线程上调用。

START

始终与 MQCTL (START) 动词同步调用。

- 在 MQCTL (START) 动词返回之前完成所有 START 回调。

与消息传递位于同一线程上 (如果请求了 THREAD_AFFINITY)。

例如, 如果先前回调在 MQCTL (START) 期间发出 MQCTL (STOP), 那么不保证具有启动的调用。

结束

在此调用之后, 将不会再传递任何消息或事件, 直到连接重新启动为止。

如果先前针对 START, 消息或事件调用了应用程序, 那么将保证 STOP。

DEREGISTER

始终是回调的最后一种调用类型。

确保应用程序在 START 和 STOP 回调中执行基于线程的初始化和清除。可以使用 REGISTER 和 DEREGISTER 回调执行基于非线程的初始化和清除。

请勿对线程的寿命和可用性进行任何非声明的假设。例如，不要依赖在上次调用 DEREGISTER 之后保持活动状态的线程。同样，当您选择不使用 THREAD_AFFINITY 时，请勿假定每当启动连接时都存在该线程。

如果应用程序对线程特征有特殊要求，那么它可以始终相应地创建线程，然后使用 MQCTL (WAIT)。这具有向 IBM WebSphere MQ "提供" 线程以进行异步消息传递的效果。

C 调用

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,  
GetMsgOpts, &CompCode, &Reason);
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,  
GETMSGOPTS, COMPCODE, REASON.
```

按如下所示声明参数:

```
** Connection handle  
01 HCONN PIC S9(9) BINARY.  
** Operation  
01 OPERATION PIC S9(9) BINARY.  
** Callback Descriptor  
01 CBDESC.  
COPY CMQCBDV.  
01 HOBJ PIC S9(9) BINARY.  
** Message Descriptor  
01 MSGDESC.  
COPY CMQMDV.  
** Get Message Options  
01 GETMSGOPTS.  
COPY CMQGMV.  
** Completion code  
01 COMPCODE PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON PIC S9(9) BINARY.
```

PL/I 调用

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,  
CompCode, Reason)
```

按如下所示声明参数:

```

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-回调函数

MQCB_FUNCTION 函数调用是用于事件处理和异步消息使用的回调函数。

仅提供 MQCB_FUNCTION 调用定义来描述传递到回调函数的参数。队列管理器未提供名为 MQCB_FUNCTION 的入口点。

要调用的实际函数的规范是 [MQCB](#) 调用的输入，并通过 [MQCBD](#) 结构传入。

语法

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Context*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。在 CICS 应用程序的 z/OS 上，以及在以兼容性方式运行的应用程序的 IBM i 上，可以省略 MQCONN 调用，并为 *Hconn* 指定以下值：

MQHC_DEF_CONN

缺省连接句柄。

MsgDesc

类型:MQMD-输入

此结构描述检索的消息的属性。

请参阅第 364 页的『MQMD - 消息描述符』，以了解详细信息。

传递的 MQMD 版本与定义使用者函数的 MQCB 调用上传递的版本相同。

如果使用版本 4 MQGMO 来请求返回消息句柄而不是 MQMD，那么 MQMD 的地址将作为空字符传递。

这是消息使用者函数的输入字段；它与事件处理程序函数无关。

GetMsg 选项

类型:MQGMO-输入

用于控制消息使用者的操作的选项。此参数还包含有关返回的消息的其他信息。

请参阅 [MQGMO](#) 以获取详细信息。

传递的 MQGMO 版本是受支持的最新版本。

这是消息使用者函数的输入字段；它与事件处理程序函数无关。

buffer

类型:MQBYTExBuffer 长度-输入

这是包含消息数据的区域。

如果没有消息可用于此调用，或者如果消息不包含消息数据，那么会将 *Buffer* 的地址作为空值传递。

这是消息使用者函数的输入字段；它与事件处理程序函数无关。

context

类型:MQCBC-输入/输出

此结构向回调函数提供上下文信息。请参阅第 250 页的『MQCBC-回调上下文』，以了解详细信息。

使用说明

1. 请注意，如果回调例程使用可能会延迟或阻塞线程的服务(例如，带有等待的 MQGET)，那么可能会延迟分派其他回调。
2. 不会为回调例程的每次调用自动建立单独的工作单元，因此例程可以发出落实调用或延迟落实，直到处理了逻辑工作批处理为止。在落实批处理工作时，它将落实自上次同步点以来调用的所有回调函数的消息。
3. 由 CICS LINK 或 CICS START 调用的程序通过指定的对象(称为通道容器)使用 CICS 服务检索参数。容器名称与参数名称相同。有关更多信息，请参阅 CICS 文档。
4. 回调例程可以发出 MQDISC 调用，但不能用于它们自己的连接。例如，如果回调例程已创建连接，那么它还可以断开连接。
5. 通常，回调例程不应依赖于每次从同一线程调用。如果需要，请在启动连接时使用 MQCTLO_THREAD_AFFINITY。
6. 当回调例程接收到非零原因码时，它必须执行相应的操作。
7. 在 IMS 适配器中不支持 MQCB_FUNCTION。

MQCLOSE-关闭对象

MQCLOSE 调用将放弃对对象的访问权，并且是 MQOPEN 和 MQSUB 调用的逆调用。

语法

MQCLOSE (*Hconn*, *Hobj*, *Options*, *CompCode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。

在 CICS 应用程序的 z/OS 上，以及在以兼容性方式运行的应用程序的 IBM i 上，可以省略 MQCONN 调用，并为 *Hconn* 指定以下值：

MQHC_DEF_HCONN

缺省连接句柄。

Hobj

类型:MQHOBJ-输入/输出

此句柄表示正在关闭的对象。对象可以是任何类型。*Hobj* 的值由先前的 MQOPEN 调用返回。

成功完成调用时，队列管理器会将此参数设置为不是环境的有效句柄的值。此值为：

MQHO_UNUSABLE_HOBJ

不可用的对象句柄。

在 z/OS 上，*Hobj* 设置为未定义的值。

options

类型:MQLONG-输入

此参数控制对象的关闭方式。

只能以多种方式关闭永久动态队列和预订，因为必须保留或删除这些队列；这些是具有 *DefinitionType* 属性的队列，其值为 MQQDT_PERMANENT_DYNAMIC (请参阅第 731 页的『队列的属性』中描述的 *DefinitionType* 属性)。本主题概述了关闭选项。

可保留或除去持久预订; 这些预订是使用带有 MQSO_持久选项的 MQSUB 调用创建的。

当关闭对受管目标的句柄 (即在使用 MQSO_MANAGED 选项的 MQSUB 调用上返回的 *Hobj* 参数) 时, 队列管理器将清除在同时除去关联预订时未检索到的任何发布。使用 MQSUB 调用上返回的 *Hsub* 参数上的 MQCO_REMOVE_SUB 选项除去预订。请注意, 对于非持久预订, MQCO_REMOVE_SUB 是 MQCLOSE 上的缺省行为。

当关闭非受管目标的句柄时, 您将负责清除发送发布的队列。首先使用 MQCO_REMOVE_SUB 关闭预订, 然后处理队列中的消息, 直到没有剩余消息为止。

必须仅从以下项指定一个选项:

动态队列选项: 这些选项控制永久动态队列的关闭方式。

MQCO_DELETE

如果以下任一项为 true, 那么将删除该队列:

- 它是由先前 MQOPEN 调用创建的永久动态队列, 并且队列上没有消息, 也没有未落实的 get 或 put 请求 (对于当前任务或任何其他任务)。
- 它是 MQOPEN 调用创建的临时动态队列, 返回了 *Hobj*。在这种情况下, 将清除队列上的所有消息。

在所有其他情况下 (包括在 MQSUB 调用上返回 *Hobj* 的情况), 调用失败, 原因码为 MQRC_OPTION_NOT_VALID_FOR_TYPE, 并且未删除对象。

在 z/OS 上, 如果该队列是逻辑上已删除的动态队列, 并且这是该队列的最后一个句柄, 那么将以物理方式删除该队列。有关更多详细信息, 请参阅第 573 页的『使用说明』。

MQCO_DELETE_PURGE

如果以下任一情况为真, 那么将删除该队列, 并清除该队列上的任何消息:

- 它是由先前 MQOPEN 调用创建的永久动态队列, 不存在未落实的获取或放入请求 (对于当前任务或任何其他任务)。
- 它是 MQOPEN 调用创建的临时动态队列, 返回了 *Hobj*。

在所有其他情况下 (包括在 MQSUB 调用上返回 *Hobj* 的情况), 调用失败, 原因码为 MQRC_OPTION_NOT_VALID_FOR_TYPE, 并且未删除对象。

| 此表显示哪些关闭选项有效, 以及是保留还是删除对象。 | | | |
|----------------------------|--------------|-------------------|--------------------------|
| 对象或队列的类型 | MQCO_NONE | MQCO_DELETE | MQCO_DELETE_PURGE |
| 队列以外的对象 | 已保留 | 无效 | 无效 |
| 预定义队列 | 已保留 | 无效 | 无效 |
| 永久动态队列 | 已保留 | 如果为空且无暂挂更新, 那么已删除 | 已删除消息; 如果没有暂挂更新, 那么队列已删除 |
| 临时动态队列 (由队列的创建者发出的调用) | 已删除 | 已删除 | 已删除 |
| 临时动态队列 (队列创建者未发出调用) | 已保留 | 无效 | 无效 |
| 通讯组列表 | 已保留 | 无效 | 无效 |
| 受管预订目标 | 已保留 | 无效 | 无效 |
| 分发列表 (已除去预订) | 已删除消息; 已删除队列 | 无效 | 无效 |

预订关闭选项: 这些选项控制关闭句柄时是否除去持久预订, 以及是否清除仍等待应用程序读取的发布。这些选项仅适用于 MQSUB 调用的 *Hsub* 参数中返回的对象句柄。

MQCO_KEEP_SUB

已关闭预订的句柄，但保留所进行的预订。继续将发布发送到预订中指定的目标。仅当使用选项 MQSO_耐久性进行预订时，此选项才有效。

如果预订是持久预订，那么 MQCO_KEEP_SUB 是缺省值

MQCO_REMOVE_SUB

将除去该预订，并关闭该预订的句柄。

MQSUB 调用的 *Hobj* 参数不会因关闭 *Hsub* 参数而失效，并且可能继续用于 MQGET 或 MQCB 以接收其余发布内容。当 MQSUB 调用的 *Hobj* 参数也关闭时，如果它是受管目标，那么将除去任何未检索的发布。

MQCO_REMOVE_SUB 是缺省值 (如果预订是非持久预订)。

下表概述了这些预订关闭选项。

要关闭持久预订句柄但保留预订，请使用以下预订关闭选项：

| 任务 | 预订关闭选项 |
|-------------------------|---------------|
| 保留 MQOPENed 句柄上的发布 | MQCO_KEEP_SUB |
| 除去 MQOPENed 句柄上的发布 | 不允许操作 |
| 将发布保留在 MQSO_MANAGED 句柄上 | MQCO_KEEP_SUB |
| 除去 MQSO_MANAGED 句柄上的发布 | 不允许操作 |

要取消预订，请通过关闭持久预订句柄并将其取消预订，或者关闭非持久预订句柄，使用以下预订关闭选项：

| 任务 | 预订关闭选项 |
|-------------------------|-----------------|
| 保留 MQOPENed 句柄上的发布 | MQCO_REMOVE_SUB |
| 除去 MQOPENed 句柄上的发布 | 不允许操作 |
| 将发布保留在 MQSO_MANAGED 句柄上 | MQCO_REMOVE_SUB |

预读选项: 以下选项控制在应用程序请求非持久消息之前已发送到客户机且尚未由应用程序使用的非持久消息发生的情况。这些消息存储在等待应用程序请求的客户机预读缓冲区中，并且可以在完成 MQCLOSE 之前从队列中废弃或使用这些消息。

MQCO_IMMEDIATE

对象将立即关闭，并且在应用程序请求之前发送到客户机的任何消息都将被废弃，并且不可供任何应用程序使用。这是缺省值。

MQCO_QUIESCE

发出了关闭对象的请求，但如果在应用程序请求之前发送到客户机的任何消息仍驻留在客户机预读缓冲区中，那么 MQCLOSE 调用将返回警告 MQRC_READ_AHEAD_MSGS，并且对象句柄保持有效。

然后，应用程序可以继续使用对象句柄来检索消息，直到不再可用为止，然后再次关闭该对象。在应用程序发出请求之前，不再向客户机发送更多消息，现在已关闭预读。

建议应用程序使用 MQCO_QUIESCE，而不是尝试到达客户机预读缓冲区中没有更多消息的点，因为消息可能在上次 MQGET 调用与以下 MQCLOSE 之间到达，如果使用了 MQCO_IMMEDIATE，那么将废弃该消息。

如果从异步回调函数中发出具有 MQCO_QUIESCE 的 MQCLOSE，那么预读消息的相同行为适用。如果返回警告 MQRC_READ_AHEAD_MSGS，那么至少再调用一次回调函数。将预读的最后一消息传递到回调函数后，MQCBC ConsumerFlags 字段将设置为 MQCBCF_READA_BUFFER_EMPTY。

缺省选项: 如果不需要上述任何选项，那么可以使用以下选项：

MQCO_NONE

不需要可选的关闭处理。

必须为以下对象指定此 必须：

- 队列以外的对象
- 预定义队列数
- 临时动态队列 (但仅在 *Hobj* 不是 创建队列的 MQOPEN 调用返回的句柄的情况下)。
- 分发列表

在上述所有情况下，将保留对象而不将其删除。

如果为临时动态队列指定了此选项：

- 如果队列是由返回 *Hobj* 的 MQOPEN 调用创建的，那么将删除该队列；将清除队列上的所有消息。
- 在所有其他情况下，将保留队列 (及其上的任何消息)。

如果为永久动态队列指定了此选项，那么将保留该队列，而不会将其删除。

在 z/OS 上，如果该队列是逻辑上已删除的动态队列，并且这是该队列的最后一个句柄，那么将以物理方式删除该队列。有关更多详细信息，请参阅 [第 573 页的『使用说明』](#)。

CompCode

类型：MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

列出的原因码是队列管理器可以针对 *Reason* 参数返回的原因码。

如果 *CompCode* 为 MQCC_OK：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING：

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') 消息组未完成。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 逻辑消息未完成。

如果 *CompCode* 为 MQCC_FAILED：

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CF_STRUC_FAILED
(2373, X'945') 耦合设施结构失败。

MQRC_CF_STRUC_IN_USE
(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CONNECTION_BROKEN
(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') 未授权连接。

MQRC_CONNECTION_STOPPING
(2203, X'89B') 连接正在关闭。

MQRC__DB2_NOT_AVAILABLE
(2342, X'926') Db2 子系统不可用。

MQRC_HCONN_ERROR
(2018, X'7E2') 连接句柄无效。

MQRC_HOBJ_ERROR
(2019, X'7E3') 对象句柄无效。

MQRC_NOT_AUTHORIZED
(2035, X'7F3') 未获得访问授权。

MQRC_OBJECT_DAMAGED
(2101, X'835') 对象已损坏。

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') 在 MQOPEN 或 MQCLOSE 调用上: 选项对于对象类型无效。

MQRC_OPTIONS_ERROR
(2046, X'7FE') 选项无效或不一致。

MQRC_PAGESET_ERROR
(2193, X'891') 访问页集数据集时出错。

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR_STOPPING
(2162, X'872') 队列管理器正在关闭。

MQRC_Q_NOT_EMPTY
(2055, X'807') 队列包含一条或多条消息或未落实的 put 或 get 请求。

MQRC_READ_AHEAD_MSGS
(nnnn, X'xxx') 客户机已预读应用程序尚未使用的消息。

MQRC_RESOURCE_PROBLEM
(2102, X'836') 没有足够系统资源可用。

MQRC_SECURITY_ERROR
(2063, X'80F') 发生了安全性错误。

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') 出口程序禁止调用。

MQRC_UNEXPECTED_ERROR
(2195, X'893') 发生了意外错误。

有关这些代码的详细信息, 请参阅[原因码](#)。

使用说明

1. 当应用程序发出 MQDISC 调用，或者正常或异常结束时，将使用 MQCO_NONE 选项自动关闭由应用程序打开且仍处于打开状态的任何对象。
2. 如果要关闭的对象是队列，那么以下要点适用：
 - 如果队列上的操作作为工作单元的一部分执行，那么可以在同步点发生之前或之后关闭该队列，而不会影响同步点的结果。如果触发队列，那么在关闭队列之前执行回滚会导致发出触发消息。有关触发器消息的更多信息，请参阅 [触发器消息的属性](#)。
 - 如果使用 MQOO_BROWSE 选项打开队列，那么将破坏浏览游标。如果随后使用 MQOO_BROWSE 选项重新打开队列，那么将创建新的浏览游标 (请参阅 [MQOO_BROWSE](#))。
 - 如果当前在 MQCLOSE 调用时对此句柄锁定了消息，那么将释放该锁定 (请参阅 [MQGMO_LOCK](#))。
 - 在 z/OS 上，如果存在针对正在关闭的队列句柄未完成 MQGMO_SET_SIGNAL 选项的 MQGET 请求，那么将取消该请求 (请参阅 [MQGMO_SET_SIGNAL](#))。针对相同队列但针对不同句柄 (*Hobj*) 提出的信号请求不受影响 (除非正在删除动态队列，在这种情况下它们也会被取消)。
3. 如果要关闭的对象是 动态队列 (永久或临时)，那么以下几点适用：

- 对于动态队列，可以指定 MQCO_DELETE 和 MQCO_DELETE_PURGE 选项，而不考虑在相应的 MQOPEN 调用上指定的选项。
- 删除动态队列时，将取消对该队列执行的所有带有 MQGMO_WAIT 选项的 MQGET 调用，并返回原因码 MQRC_Q_DELETED。请参阅 [MQGMO_WAIT](#)。

虽然应用程序无法访问已删除的队列，但不会从系统中除去该队列，并且不会释放关联的资源，直到引用该队列的所有句柄都已关闭，并且影响该队列的所有工作单元都已落实或回退。

在 z/OS 上，已逻辑删除但尚未从系统中除去的队列会阻止创建与已删除的队列同名的新队列；在这种情况下，MQOPEN 调用将失败，原因码为 MQRC_NAME_IN_USE。此外，仍可以使用 MQSC 命令显示此类队列，即使应用程序无法访问此队列也是如此。

- 删除永久动态队列时，如果在 MQCLOSE 调用上指定的 *Hobj* 句柄不是创建队列的 MQOPEN 调用返回的句柄，那么将检查用于验证 MQOPEN 调用的用户标识是否有权删除该队列。如果在 MQOPEN 调用上指定了 MQOO_ALTERNATE_USER_AUTHORITY 选项，那么检查的用户标识为 *AlternateUserId*。

在下列情况下，将不执行此检查：

- 指定的句柄是创建队列的 MQOPEN 调用返回的句柄。
 - 正在删除的队列是临时动态队列。
- 当临时动态队列关闭时，如果 MQCLOSE 调用上指定的 *Hobj* 句柄是创建队列的 MQOPEN 调用返回的句柄，那么将删除该队列。无论 MQCLOSE 调用上指定的关闭选项如何，都会发生此情况。如果队列中有消息，那么将废弃这些消息；不会生成任何报告消息。

如果存在影响队列的未落实工作单元，那么仍会删除队列及其消息，但工作单元不会失败。但是，如上所述，在落实或回退每个工作单元之前，不会释放与工作单元相关联的资源。

4. 如果要关闭的对象是 分发列表，那么以下要点适用：
 - 分发列表的唯一有效关闭选项是 MQCO_NONE；调用失败，原因码为 MQRC_OPTIONS_ERROR 或 MQRC_OPTION_NOT_VALID_FOR_TYPE (如果指定了任何其他选项)。
 - 当分发列表关闭时，不会针对列表中的队列返回个别完成代码和原因码；只有调用的 *CompCode* 和 *Reason* 参数可用于诊断目的。

如果关闭其中一个队列时发生故障，那么队列管理器将继续处理并尝试关闭分发列表中的其余队列。调用的 *CompCode* 和 *Reason* 参数设置为返回描述故障的信息。完成代码可能为 MQCC_FAILED，即使大多数队列已成功关闭也是如此。未识别迁到错误的队列。

如果在多个队列上发生故障，那么未定义在 *CompCode* 和 *Reason* 参数中报告的故障。
5. 在 IBM i 上，如果发出第一个 MQOPEN 调用时已隐式连接应用程序，那么发出最后一个 MQCLOSE 时将发生隐式 MQDISC。

只有以兼容性方式运行的应用程序才能隐式连接;其他应用程序必须发出 MQCONN 或 MQCONNX 调用以显式连接到队列管理器。

C 调用

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.
```

PL/I 调用

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

按如下所示声明参数:

```
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 */
```

高级汇编程序调用

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

按如下所示声明参数:

```
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
```

Visual Basic 调用

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

按如下所示声明参数:

```
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-落实更改

MQCMIT 调用向队列管理器指示应用程序已到达同步点, 并且自上次同步点以来发生的所有消息获取和放置都将成为永久消息。

作为工作单元的一部分放入的消息可供其他应用程序使用; 作为工作单元的一部分检索的消息将被删除。

- 在 z/OS 上, 调用仅由批处理程序 (包括 IMS 批处理 DL/I 程序) 使用。
- 在 IBM i 上, 以兼容性方式运行的应用程序不支持此调用。

语法

MQCMIT (*Hconn*, *CompCode*, *Reason*)

参数

Hconn

类型: MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

列出的原因码是队列管理器可以针对 *Reason* 参数返回的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_BACKED_OUT

(2003, X'7D3') 工作单元已回退。

MQRC_OUTCOME_PENDING

(2124, X'84C') 落实操作的结果处于暂挂状态。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CALL_中断

(2549, X'9F5') MQPUT 或 MQCMIT 已中断, 重新连接处理无法重新建立明确的结果。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') 调用在环境中无效。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_OBJECT_DAMAGED

(2101, X'835 ') 对象已损坏。

MQRC_OUTCOME_MIXED

(2123, X'84B') 落实或回退操作的结果是混合的。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RECONNECT_FAILED

(2548, X'9F4') 重新连接后, 恢复可重新连接的连接的句柄时发生错误。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') 外部存储介质已满。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息, 请参阅原因码。

使用说明

1. 仅当队列管理器本身协调工作单元时, 才使用此调用。这可以是:

- 本地工作单元, 其中的更改仅影响 WebSphere MQ 资源。
- 全局工作单元, 其中的更改可能会影响属于其他资源管理器的资源, 也会影响 WebSphere MQ 资源。

有关本地和全局工作单元的更多详细信息, 请参阅第 552 页的『MQBEGIN-开始工作单元』。

2. 在队列管理器未协调工作单元的环境中, 必须使用相应的落实调用来代替 MQCMIT。该环境还可能支持由应用程序正常终止而导致的隐式落实。

• 在 z/OS 上, 使用以下调用:

- 如果工作单元仅影响 WebSphere MQ 资源, 那么批处理程序 (包括 IMS 批处理 DL/I 程序) 可以使用 MQCMIT 调用。但是, 如果工作单元同时影响 WebSphere MQ 资源和属于其他资源管理器的资源 (例如, DB2), 请使用 z/OS 可恢复资源服务 (RRS) 提供的 SRRCMIT 调用。SRRCMIT 调用会将更改落实到属于已启用 RRS 协调的资源管理器的资源。

- CICS 应用程序必须使用 EXEC CICS SYNCPOINT 命令来显式落实工作单元。或者，结束事务会导致隐式落实工作单元。MQCMIT 调用不能用于 CICS 应用程序。
 - IMS 应用程序 (批处理 DL/I 程序除外) 必须使用 IMS 调用 (例如 GU 和 CHKP) 来落实工作单元。MQCMIT 调用不能用于 IMS 应用程序 (批处理 DL/I 程序除外)。
 - 在 IBM i 上，将此调用用于队列管理器协调的本地工作单元。这意味着在作业级别不得存在落实定义，即，不得对作业发出带有 CMTSCOPE(*JOB) 参数的 STRCMTCTL 命令。
3. 如果应用程序在工作单元中以未落实的更改结束，那么这些更改的处置取决于应用程序是正常结束还是异常结束。请参阅 MQDISC 使用说明 以获取更多详细信息。
 4. 当应用程序在逻辑消息的组或段中放置或获取消息时，队列管理器会保留与消息组和上次成功 MQPUT 和 MQGET 调用的逻辑消息相关的信息。此信息与队列句柄相关联，并包括如下内容：
 - MQMD 中 *GroupId*, *MsgSeqNumber*, *Offset* 和 *MsgFlags* 字段的值。
 - 消息是否是工作单元的一部分。
 - 对于 MQPUT 调用: 消息是持久消息还是非持久消息。

落实工作单元时，队列管理器会保留组和段信息，并且应用程序可以继续将消息放入当前消息组或逻辑消息中或获取消息。

在落实工作单元时保留组和段信息允许应用程序在多个工作单元之间传播由多个段组成的大型消息组或大型逻辑消息。如果本地队列管理器只有有限的队列存储器，那么使用多个工作单元是有利的。但是，如果发生系统故障，应用程序必须保留足够的信息以重新启动在正确位置放置或获取消息。有关在系统故障后如何在正确位置重新启动的详细信息，请参阅 [MQPMO_LOGICAL_ORDER](#) 和 [MQGMO_LOGICAL_ORDER](#)。

仅当队列管理器协调工作单元时，其余使用说明才适用：

5. 工作单元具有与连接句柄相同的作用域; 必须使用同一连接句柄来执行影响特定工作单元的所有 WebSphere MQ 调用。使用另一个连接句柄发出的调用 (例如，另一个应用程序发出的调用) 会影响另一个工作单元。请参阅 MQCONN 中描述的 *Hconn* 参数，以获取有关连接句柄作用域的信息。
6. 只有作为当前工作单元的一部分放入或检索的消息才会受此调用影响。
7. 在工作单元中发出 MQGET，MQPUT 或 MQPUT1 调用但从不出落实或回退调用的长时间运行的应用程序可以使用不可用于其他应用程序的消息填充队列。要防止发生此情况，管理员必须将 *MaxUncommittedMsgs* 队列管理器属性设置为足以防止失控应用程序填充队列的值，但设置为足以允许期望的消息传递应用程序正确工作的值。
8. 在 UNIX 和 Windows 系统上，如果 *Reason* 参数为 MQRC_CONNECTION_BROKEN (*CompCode* 为 MQCC_FAILED) 或 MQRC_UNEXPECTED_ERROR，那么可能已成功落实工作单元。

C 调用

```
MQCMIT (Hconn, &CompCode, &Reason);
```

按如下所示声明参数：

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

COBOL 调用

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

按如下所示声明参数：

```
** 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.
```

PL/I 调用

```
call MQCMIT (Hconn, CompCode, Reason);
```

按如下所示声明参数:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

高级汇编程序调用

```
CALL MQCMIT,(HCONN,COMPCODE,REASON)
```

按如下所示声明参数:

```
HCONN      DS  F  Connection handle
COMPCODE   DS  F  Completion code
REASON     DS  F  Reason code qualifying COMPCODE
```

Visual Basic 调用

```
MQCMIT Hconn, CompCode, Reason
```

按如下所示声明参数:

```
Dim Hconn      As Long 'Connection handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'
```

MQCONN - 连接队列管理器

MQCONN 调用可将应用程序连接到队列管理器。

它可提供要给队列管理器连接句柄，以供应用程序在后续消息排队调用上使用。

- 在 z/OS 上，CICS 应用程序不必发出此调用。这些应用程序会自动连接到 CICS 系统所连接的队列管理器。但是，仍接受来自 CICS 应用程序的 MQCONN 和 MQDISC 调用。
- 在 IBM i 上，以兼容性方式运行的应用程序不必发出此调用。这些应用程序在发出第一个 MQOPEN 调用时自动连接到队列管理器。但是，仍接受来自 IBM i 应用程序的 MQCONN 和 MQDISC 调用。

其他应用程序(即，未以兼容性方式运行的应用程序)必须使用 MQCONN 或 MQCONNX 调用来连接到队列管理器，并使用 MQDISC 调用来断开与队列管理器的连接。这是推荐的编程风格。

无法在纯服务器安装中进行客户机连接，并且无法在纯客户机安装中进行本地连接。

语法

```
MQCONN (QMgrName, Hconn, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

这是应用程序要连接到的队列管理器的名称。此名称可包含以下字符:

- 大写字母字符 (A 到 Z)
- 小写字母字符 (a 到 z)
- 数字位 (0 到 9)
- 句点 (.)、正斜杠 (/)、下划线 (_)、百分号 (%)

此名称不得包含前导空格或嵌入空格,但可以包含尾部空格。空字符可用于指示名称中重要数据结束;空字符及空字符后的任何字符都作为空格来处理。以下限制适用于指示的环境中:

- 在使用 EBCDIC 片假名的系统上,不得使用小写字符。
- 在 z/OS 上,操作和控制面板无法处理以下划线开头或结尾的名称。因此,请避免使用此类名称。
- 在 IBM i 上,当在命令上指定时,将包含小写字符,正斜杠或百分号的名称括在引号中。请勿在 *QMgrName* 参数中指定这些引号。

如果名称完全由空格组成,那么将使用缺省队列管理器的名称。

针对 *QMgrName* 指定的名称必须为可连接的队列管理器的名称。

在 z/OS 上,可以连接到的队列管理器由环境确定:

- 对于 CICS,只能使用 CICS 系统所连接的队列管理器。仍必须指定 *QMgrName* 参数,但忽略其值;建议使用空白。
- 对于 IMS,只有在子系统定义表 (CSQQDEFV) 中列出的队列管理器和 (在 IMS 的 SSM 表中列出) 可连接 (请参阅用法说明 6)。
- 对于 z/OS 批处理和 TSO,只有与应用程序位于同一系统上的队列管理器可连接 (请参阅用法说明 6)。

队列共享组: 在存在多个队列管理器并且这些队列管理器已配置为组成队列共享组的系统上,可针对 *QMgrName* 指定队列共享组的名称以代替队列管理器的名称。这允许应用程序连接到队列共享组中可用的任何队列管理器以及与应用程序位于同一 z/OS 映像上的队列管理器。还可以将此系统配置为使用空白 *QMgrName* 来连接到队列共享组而不是缺省队列管理器。

如果 *QMgrName* 指定了队列共享组的名称,但在系统上还存在具有相同名称的队列管理器,那么将优先连接到后者的队列管理器。仅当此连接失败后,才会尝试连接到队列共享组中的某个队列管理器。

如果连接成功,那么您可以使用由 MQCONN 或 MQCONNX 调用返回的句柄来访问属于已连接到的队列管理器的所有资源 (包括已共享和未共享的资源)。对这些资源的访问受到典型授权控件的影响。

如果应用程序发出两个 MQCONN 或 MQCONNX 调用以建立并行连接,并且其中一个调用或者两个调用都指定队列共享组的名称,那么第二个调用在连接到与第一个调用相同的队列管理器时会返回完成代码 MQCC_WARNING 和原因码 MQRC_ALREADY_CONNECTED。

仅在 z/OS 上支持队列共享组。仅在批处理, RRS 批处理和 TSO 环境中支持与队列共享组的连接。

WebSphere MQ MQI 客户机应用程序: 对于 WebSphere MQ MQI 客户机应用程序,将尝试对每个具有指定队列管理器名称的客户机连接通道定义进行连接,直到成功为止。但此队列管理器的名称必须与指定名称相同。如果指定全空白名称,那么将尝试具有全空白队列管理器名称的每个客户机连接通道,直到有一个连接成功为止;在此情况下,针对队列管理器的实际名称不执行任何检查。

WebSphere MQ 客户机应用程序在 z/OS 中不受支持,但是 z/OS 可以充当 WebSphere MQ 客户机应用程序可以连接到的 WebSphere MQ 服务器。

WebSphere MQ MQI 客户机队列管理器组: 如果指定的名称以星号 (*) 开头,那么与其建立连接的队列管理器可能具有与应用程序指定的名称不同的名称。指定的名称 (不含星号) 可定义一组可连接的队列管理器。此实施会从组中选择一个队列管理器,方法是轮流尝试每个队列管理器,直到找到可连接的队列管理器为止。尝试连接的顺序受到客户机通道权重和候选通道的连接亲缘关系值的影响。如果组中没有任何队列管理器可供连接,那么调用失败。每个队列管理器仅尝试一次。如果针对名称仅指定了一个星号,那么将使用实施定义的缺省队列管理器组。

仅针对 MQ 客户机环境中运行的应用程序才支持队列管理器组；如果非客户机应用程序指定以星号开头的队列管理器名称，那么调用将失败。通过提供多个具有相同队列管理器名称（不含星号的指定名称）的客户机连接通道定义来定义组，以与组中每个队列管理器进行通信。通过提供一个或多个客户机连接通道定义来定义缺省组，其中每个都具有空白的队列管理器名称（指定全空白名称从而实现与针对每个客户机应用程序名称指定一个星号相同的效果）。

在连接到组的一个队列管理器之后，应用程序可以在消息和对象描述符中的队列管理器名称字段中以典型方式指定空白，以表示应用程序已连接到的队列管理器（本地队列管理器）的名称。如果此应用程序需要知道此名称，请使用 MQINQ 调用来查询 *QMGrName* 队列管理器属性。

在连接名称中使用星号作为前缀表示应用程序不依赖于连接到组中的特定队列管理器。适合的应用程序为：

- 放置消息但不获取消息的应用程序。
- 放置请求消息然后从临时动态队列中获取应答消息的应用程序。

不适合的应用程序为需要从特定队列管理器中的特定队列获取消息的应用程序；此类应用程序不得使用星号作为其名称的前缀。

如果指定星号，那么此名称的剩余部分的最大长度为 47 个字符。

在 z/OS 上不支持队列管理器组。

此参数的长度由 MQ_Q_MGR_NAME_LENGTH 提供。

Hconn

类型：MQHCONN - 输出

此句柄表示与队列管理器的连接。在由应用程序发出的所有后续消息排队调用上指定此句柄。发出 MQDISC 调用时，或者当定义句柄作用域的处理单元终止时，此句柄将不再有效。

WebSphere MQ 现在向 mqm 库提供客户机软件包以及服务器软件包。这意味着执行 mqm 库中找到的 MQI 调用时，将检查连接类型以查看它属于客户机连接还是服务器连接，然后执行正确的底层调用。因此，现在可针对 mqm 库链接传递 *Hconn* 的出口，但在客户机安装上使用此出口。

句柄作用域：返回的句柄作用域取决于用于连接到队列管理器的调用（MQCONN 或 MQCONNX）。如果使用的调用为 MQCONNX，那么此句柄的作用域还取决于 MQCNO 结构的 *Options* 字段中指定的 MQCNO_HANDLE_SHARE_* 选项。

- 如果调用为 MQCONN，或者如果指定了 MQCNO_HANDLE_SHARE_NONE 选项，那么返回的句柄为非共享句柄。

非共享句柄的作用域是运行应用程序的平台所支持的最小并行处理单元（请参阅 [第 580 页的表 564](#) 以获取详细信息）；句柄在发出调用的并行处理单元之外无效。

- 如果指定 MQCNO_HANDLE_SHARE_BLOCK 或 MQCNO_HANDLE_SHARE_NO_BLOCK 选项，那么返回的句柄为共享句柄。

共享句柄的作用域为拥有从中发出调用的线程的进程；可从属于该进程的任何线程使用此句柄。并非所有平台都支持线程。

- 如果 MQCONN 或 MQCONNX 调用失败，并且完成代码为 MQCC_FAILED，那么 Hconn 值未定义。

| 平台 | 非共享句柄的作用域 |
|---------|--|
| z/OS | <ul style="list-style-type: none"> • CICS: CICS 任务 • IMS: 任务，直到下一个同步点（不包括任务的子任务） • z/OS 批处理和 TSO: 任务（不包括任务的子任务） |
| IBM i | 作业 |
| UNIX 系统 | 线程 |

| 表 564: 各种平台上非共享句柄的作用域 (继续) | |
|----------------------------|-----------|
| 平台 | 非共享句柄的作用域 |
| 16 位 Windows 应用程序 | 进程 |
| 32 位 Windows 应用程序 | 线程 |

在 z/OS for CICS 应用程序和 IBM i for applications in compatibility mode , 返回的值为:

MQHC_DEF_HCONN

缺省连接句柄。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_ALREADY_CONNECTED

(2002, X'7D2') 应用程序已连接。

MQRC_CLUSTER_EXIT_LOAD_ERROR

(2267, X'8DB') 无法装入集群工作负载出口。

MQRC_SSL_ALREADY_INITIALIZED

(2391, X'957') 已初始化 SSL。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR

(2129, X'851') 无法装入适配器连接模块。

MQRC_ADAPTER_DEFS_ERROR

(2131, X'853') 适配器子系统定义模块无效。

MQRC_ADAPTER_DEFS_LOAD_ERROR

(2132, X'854') 无法装入适配器子系统定义模块。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ADAPTER_STORAGE_SHORTAGE

(2127, X'84F') 适配器存储空间不足。

MQRC_ANOTHER_Q_MGR_CONNECTED

(2103, X'837') 已连接另一个队列管理器。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_INIT_ERROR

(2375, X'947') API 出口初始化失败。

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') API 出口终止失败。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CONN_ID_IN_USE

(2160, X'870') 连接标识已在使用中。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_ERROR

(2273, X'8E1') 处理 MQCONN 调用时出错。

MQRC_CONNECTION_NOT_AVAILABLE

(2568, X'A08') 当队列管理器无法在当前安装中提供所请求连接类型的连接时，在 MQCONN 或 MQCONNX 调用上发生此情况。无法在纯服务器安装中进行客户机连接。无法在纯客户机安装中进行本地连接。

MQRC_CONNECTION_QUIESCING

(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_CRYPTO_HARDWARE_ERROR

(2382, X'94E') 加密硬件配置错误。

MQRC_DUPLICATE_RECOV_COORD

(2163, X'873') 恢复协调程序已存在。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') 调用在环境中无效。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') 从客户机发出了 MQCONN 调用以连接到队列管理器，但尝试将对话分配给远程系统失败。

MQRC_INSTALLATION_MISMATCH

(2583, X'A17') 在队列管理器安装与所选库之间存在不匹配。

MQRC_KEY_REPOSITORY_ERROR

(2381, X'94D') 密钥存储库无效。

MQRC_MAX_CONNS_LIMIT_REACHED

(2025, X'7E9') 已达到最大连接数。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_OPEN_FAILED

(2137, X'859') 对象未成功打开。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING

(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_SECURITY_ERROR

(2063, X'80F') 发生了安全性错误。

MQRC_SSL_INITIALIZATION_ERROR

(2393, X'959') SSL 初始化错误。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅原因码。

使用说明

1. 使用 MQCONN 调用连接到的队列管理器称为本地队列管理器。
2. 由本地队列管理器所拥有的队列对于该应用程序显示为本地队列。可以在这些队列中放置消息和获取消息。

由本地队列管理器所属的队列共享组所拥有的共享队列对于该应用程序显示为本地队列。可以在这些队列中放置消息和获取消息。

由远程队列管理器所拥有的队列显示为远程队列。可以在这些队列上放置消息，但是不能从中获取消息。

3. 如果队列管理器在应用程序运行时失败，那么应用程序必须再次发出 MQCONN 调用以获取要在后续 WebSphere MQ 调用上使用的新连接句柄。此应用程序可以定期发出 MQCONN 调用，直至调用成功为止。

如果应用程序不确定是否已连接到队列管理器，那么该应用程序可以安全地发出 MQCONN 调用以获取连接句柄。如果该应用程序已连接，那么返回的句柄与先前 MQCONN 调用返回的句柄相同，但会显示完成代码 MQCC_WARNING 和原因码 MQRC_ALREADY_CONNECTED。

4. 当应用程序完成使用 WebSphere MQ 调用时，应用程序必须使用 MQDISC 调用与队列管理器断开连接。
5. 如果 MQCONN 调用失败，并且完成代码为 MQCC_FAILED，那么 Hconn 值未定义。
6. 在 z/OS 上：

- 批处理，TSO 和 IMS 应用程序必须发出 MQCONN 调用以使用其他 WebSphere MQ 调用。这些应用程序可同时连接到多个队列管理器。

如果队列管理器发生故障，那么队列管理器重新启动后，该应用程序必须重新发出此调用以获取新的连接句柄。

尽管 IMS 应用程序可以重复发出 MQCONN 调用（即使已连接），但不建议对联机消息处理程序（MPP）执行此操作。

- CICS 应用程序不必发出 MQCONN 调用以使用其他 WebSphere MQ 调用，但可以根据需要执行此操作；同时接受 MQCONN 调用和 MQDISC 调用。但是，不能同时连接到多个队列管理器。

如果队列管理器发生故障，那么队列管理器重新启动时，这些应用程序会自动重新连接，因此无需发出 MQCONN 调用。

7. 在 z/OS 上，要定义可用的队列管理器：

- 对于批处理应用程序，系统程序员可以使用 CSQBDEF 宏来创建模块（CSQBDEFV），此模块可定义缺省队列管理器名称或队列共享组名称。

- 对于 IMS 应用程序，系统程序员可以使用 CSQQDEFX 宏来创建模块 (CSQQDEFV)，以定义可用队列管理器的名称并指定缺省队列管理器。

此外，必须将每个队列管理器定义到 IMS 控制区域以及访问该队列管理器的每个从属区域。为此，必须在 IMS 中创建子系统成员。PROCLIB 库并向适用的 IMS 区域标识子系统成员。如果应用程序尝试连接到未在其 IMS 区域的子系统成员中定义的队列管理器，那么应用程序将异常终止。

8. 在 IBM i 上，可以运行行为队列管理器的先前发行版编写的应用程序，而无需重新编译。这叫兼容性方式。此操作方式为应用程序提供兼容的运行环境。它包括以下内容：

- 驻留在库 QMQM 中的服务程序 AMQZSTUB。

AMQZSTUB 提供与先前发行版相同的公共接口，并且具有相同的签名。使用此服务程序通过绑定过程调用访问 MQI。

- 驻留在库 QMQM 中的程序 QMQM。

QMQM 提供了通过动态程序调用访问 MQI 的方法。

- 驻留在库 QMQM 中的程序 MQCLOSE，MQCONN，MQ 光盘，MQGET，MQINQ，MQOPEN，MQPUT，MQPUT1 和 MQSET。

这些程序还提供了通过动态程序调用访问 MQI 的方法，但具有与 WebSphere MQ 调用的标准描述相对应的参数列表。

这三个接口不包含 WebSphere MQ V 5.1 中引入的功能。例如，不支持 MQBACK，MQCMIT 和 MQCONNX 调用。这些接口提供的支持仅适用于单线程应用程序。

通过服务程序 LIBMQM 和 LIBMQM_R 提供了对单线程应用程序中的新 WebSphere MQ 调用的支持，以及对多线程应用程序中的所有 WebSphere MQ 调用的支持。

9. 在 IBM i 上，异常结束的程序不会自动与队列管理器断开连接。请编写应用程序以允许 MQCONN 或 MQCONNX 调用返回完成代码 MQCC_WARNING 和原因码 MQRC_ALREADY_CONNECTED。在此情况下，请正常使用返回的连接句柄。

C 调用

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

按如下所示声明参数：

```
MQCHAR48 QMgrName; /* Name of queue manager */
MQHCONN  Hconn;    /* Connection handle */
MQLONG   CompCode; /* Completion code */
MQLONG   Reason;   /* Reason code qualifying CompCode */
```

COBOL 调用

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

按如下所示声明参数：

```
** 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.
```


PL/I 调用

```
call MQCONN (QMGrName, Hconn, CompCode, Reason);
```

按如下所示声明参数:

```
dc1 QMGrName char(48);      /* Name of queue manager */
dc1 Hconn    fixed bin(31); /* Connection handle */
dc1 CompCode fixed bin(31); /* Completion code */
dc1 Reason   fixed bin(31); /* Reason code qualifying CompCode */
```

高级汇编程序调用

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

按如下所示声明参数:

```
QMGRNAME DS CL48 Name of queue manager
HCONN     DS F   Connection handle
COMPCODE  DS F   Completion code
REASON    DS F   Reason code qualifying COMPCODE
```

Visual Basic 调用

```
MQCONN QMGrName, Hconn, CompCode, Reason
```

按如下所示声明参数:

```
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-连接队列管理器 (扩展)

MQCONNX 调用将应用程序连接到队列管理器。它提供队列管理器连接句柄，供应用程序在后续 WebSphere MQ 调用上使用。

MQCONNX 调用类似于 MQCONN 调用，只是 MQCONNX 允许指定选项来控制调用的工作方式。

- 此调用在所有 WebSphere MQ 系统以及连接到这些系统的 WebSphere MQ 客户机上受支持。
- 在 IBM i 上，以兼容性方式运行的应用程序不支持此调用。

无法在纯服务器安装中进行客户机连接，并且无法在纯客户机安装中进行本地连接。

语法

```
MQCONNX (QMGrName, ConnectOpts, Hconn, CompCode, Reason)
```

参数

QMGrName

类型: MQCHAR48 - 输入

请参阅第 578 页的『MQCONN - 连接队列管理器』中描述的 QMGrName 参数以获取详细信息。

ConnectOpts

类型:MQCNO-输入/输出

请参阅第 283 页的『MQCNO - 连接选项』，以了解详细信息。

Hconn

类型:MQHCONN - 输出

此句柄表示与队列管理器的连接。在由应用程序发出的所有后续消息排队调用上指定此句柄。发出 MQDISC 调用时，或者当定义句柄作用域的处理单元终止时，此句柄将不再有效。

WebSphere MQ 现在向 mqm 库提供客户机软件包以及服务器软件包。这意味着执行 mqm 库中找到的 MQI 调用时，将检查连接类型以查看它属于客户机连接还是服务器连接，然后执行正确的底层调用。因此，现在可针对 mqm 库链接传递 Hconn 的出口，但在客户机安装上使用此出口。

句柄作用域：返回的句柄作用域取决于用于连接到队列管理器的调用（MQCONN 或 MQCONNX）。如果使用的调用为 MQCONNX，那么此句柄的作用域还取决于 MQCNO 结构的 *Options* 字段中指定的 MQCNO_HANDLE_SHARE_* 选项。

- 如果调用为 MQCONN，或者如果指定了 MQCNO_HANDLE_SHARE_NONE 选项，那么返回的句柄为非共享句柄。

非共享句柄的作用域是运行应用程序的平台所支持的最小并行处理单元 (请参阅第 586 页的表 565 以获取详细信息); 句柄在发出调用的并行处理单元之外无效。

- 如果指定 MQCNO_HANDLE_SHARE_BLOCK 或 MQCNO_HANDLE_SHARE_NO_BLOCK 选项，那么返回的句柄为共享句柄。

共享句柄的作用域为拥有从中发出调用的线程的进程; 可从属于该进程的任何线程使用此句柄。并非所有平台都支持线程。

- 如果 MQCONN 或 MQCONNX 调用失败，并且完成代码为 MQCC_FAILED，那么 Hconn 值未定义。

| 平台 | 非共享句柄的作用域 |
|-------------------|--|
| z/OS | <ul style="list-style-type: none">• CICS: CICS 任务• IMS: 任务，直到下一个同步点 (不包括任务的子任务)• z/OS 批处理和 TSO: 任务 (不包括任务的子任务) |
| IBM i | 作业 |
| UNIX 系统 | 线程 |
| 16 位 Windows 应用程序 | 进程 |
| 32 位 Windows 应用程序 | 线程 |

在 z/OS for CICS 应用程序和 IBM i for applications in compatibility mode , 返回的值为:

MQHC_DEF_HCONN

缺省连接句柄。

CompCode

类型: MQLONG - 输出

请参阅第 578 页的『MQCONN - 连接队列管理器』中描述的 *CompCode* 参数以获取详细信息。

原因

类型: MQLONG - 输出

MQCONN 和 MQCONNX 调用可返回以下代码。有关 MQCONNX 调用可返回的其他代码的列表，请参阅以下代码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_ALREADY_CONNECTED

(2002, X'7D2') 应用程序已连接。

MQRC_CLUSTER_EXIT_LOAD_ERROR

(2267, X'8DB') 无法装入集群工作负载出口。

MQRC_SSL_ALREADY_INITIALIZED

(2391, X'957') 已初始化 SSL。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR

(2129, X'851') 无法装入适配器连接模块。

MQRC_ADAPTER_DEFS_ERROR

(2131, X'853') 适配器子系统定义模块无效。

MQRC_ADAPTER_DEFS_LOAD_ERROR

(2132, X'854') 无法装入适配器子系统定义模块。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ADAPTER_STORAGE_SHORTAGE

(2127, X'84F') 适配器存储空间不足。

MQRC_ANOTHER_Q_MGR_CONNECTED

(2103, X'837') 已连接另一个队列管理器。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_INIT_ERROR

(2375, X'947') API 出口初始化失败。

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') API 出口终止失败。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CONN_ID_IN_USE

(2160, X'870') 连接标识已在使用中。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_ERROR

(2273, X'8E1') 处理 MQCONN 调用时出错。

MQRC_CONNECTION_NOT_AVAILABLE

(2568, X'A08') 当队列管理器无法在当前安装中提供所请求连接类型的连接时，在 MQCONN 或 MQCONNX 调用上发生此情况。无法在纯服务器安装中进行客户机连接。无法在纯客户机安装中进行本地连接。

MQRC_CONNECTION_QUIESCING

(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_CRYPTO_HARDWARE_ERROR

(2382, X'94E') 加密硬件配置错误。

MQRC_DUPLICATE_RECOV_COORD

(2163, X'873') 恢复协调程序已存在。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') 调用在环境中无效。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') 从客户机发出了 MQCONN 调用以连接到队列管理器，但尝试将对话分配给远程系统失败。

MQRC_INSTALLATION_MISMATCH

(2583, X'A17') 在队列管理器安装与所选库之间存在不匹配。

MQRC_KEY_REPOSITORY_ERROR

(2381, X'94D') 密钥存储库无效。

MQRC_MAX_CONNS_LIMIT_REACHED

(2025, X'7E9') 已达到最大连接数。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_OPEN_FAILED

(2137, X'859') 对象未成功打开。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING

(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_SECURITY_ERROR

(2063, X'80F') 发生了安全性错误。

MQRC_SSL_INITIALIZATION_ERROR

(2393, X'959') SSL 初始化错误。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

MQCONN 调用可返回以下其他原因码：

如果 *CompCode* 为 MQCC_FAILED：

MQRC_AIR_ERROR

(2385, X'951') 认证信息记录无效。

MQRC_AUTH_INFO_CONN_NAME_ERROR

(2387, X'953') 认证信息连接名称无效。

MQRC_AUTH_INFO_REC_COUNT_ERROR

(2383, X'94F') 认证信息记录计数无效。

MQRC_AUTH_INFO_REC_ERROR

(2384, X'950') 认证信息记录字段无效。

MQRC_AUTH_INFO_TYPE_ERROR

(2386, X'952') 认证信息类型无效。

MQRC_CD_ERROR

(2277, X'8E5') 通道定义无效。

MQRC_CLIENT_CONN_ERROR

(2278, X'8E6') 客户机连接字段无效。

MQRC_CNO_ERROR

(2139, X'85B') Connect-options 结构无效。

MQRC_CONN_TAG_IN_USE

(2271, X'8DF') 连接标记正在使用中。

MQRC_CONN_TAG_NOT_USABLE

(2350, X'92E') 连接标记不可用。

MQRC_LDAP_PASSWORD_ERROR

(2390, X'956') LDAP 密码无效。

MQRC_LDAP_USER_NAME_ERROR

(2388, X'954') LDAP 用户名字段无效。

MQRC_LDAP_USER_NAME_LENGTH_ERR

(2389, X'955') LDAP 用户名长度无效。

MQRC_OPTIONS_ERROR

(2046, X'7FE') 选项无效或不一致。

MQRC_SCO_ERROR

(2380, X'94C') SSL 配置选项结构无效。

MQRC_SSL_CONFIG_ERROR

(2392, X'958') SSL 配置错误。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

对于 Visual Basic 编程语言，以下要点适用：

- *ConnectOpts* 参数声明为 MQCNO 类型。如果应用程序正在作为 WebSphere MQ MQI 客户机运行，并且您想要指定客户机连接通道的参数，请将 *ConnectOpts* 参数声明为 Any 类型，以便应用程序可以在调用上指定 MQCNOCD 结构来代替 MQCNO 结构。但是，这意味着无法检查 *ConnectOpts* 参数以确保它是正确的数据类型。

C 调用

```
MQCONN (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

按如下所示声明参数：

```
MQCHAR48  QMgrName;      /* Name of queue manager */
MQCNO     ConnectOpts;  /* Options that control the action of MQCONN */
MQHCONN   Hconn;        /* Connection handle */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

COBOL 调用

```
CALL 'MQCONN' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,  
REASON.
```

按如下所示声明参数:

```
** Name of queue manager  
01 QMGRNAME PIC X(48).  
** Options that control the action of MQCONN  
01 CONNECTOPTS.  
COPY CMQCNV.  
** 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.
```

PL/I 调用

```
call MQCONN (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

按如下所示声明参数:

```
dcl QMgrName char(48); /* Name of queue manager */  
dcl ConnectOpts like MQCNO; /* Options that control the action of  
MQCONN */  
dcl Hconn fixed bin(31); /* Connection handle */  
dcl CompCode fixed bin(31); /* Completion code */  
dcl Reason fixed bin(31); /* Reason code qualifying CompCode */
```

高级汇编程序调用

```
CALL MQCONN,(QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

按如下所示声明参数:

```
QMGRNAME DS CL48 Name of queue manager  
CONNECTOPTS CMQCNVA , Options that control the action of MQCONN  
HCONN DS F Connection handle  
COMPCODE DS F Completion code  
REASON DS F Reason code qualifying COMPCODE
```

Visual Basic 调用

```
MQCONN QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

按如下所示声明参数:

```
Dim QMgrName As String*48 'Name of queue manager'  
Dim ConnectOpts As MQCNO 'Options that control the action of'  
'MQCONN'  
Dim Hconn As Long 'Connection handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCRTMH-创建消息句柄

MQCRTMH 调用返回消息句柄。

应用程序可以在后续消息排队调用上使用 MQCRTMH 调用：

- 使用 [MQSETMP](#) 调用来设置消息句柄的属性。
- 使用 [MQINQMP](#) 调用来查询消息句柄的属性值。
- 使用 [MQDLTMP](#) 调用来删除消息句柄的属性。

可以在 MQPUT 和 MQPUT1 调用上使用消息句柄，以将消息句柄的属性与正在放入的消息的属性相关联。同样，通过在 MQGET 调用上指定消息句柄，可以在 MQGET 调用完成时使用消息句柄来访问正在检索的消息的属性。

使用 [MQDLTMH](#) 来删除消息句柄。

语法

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Reason*)

参数

Hconn

类型 :MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。如果与队列管理器的连接不再有效，并且没有 WebSphere MQ 调用在消息句柄上运行，那么将隐式调用 [MQDLTMH](#) 以删除消息。

或者，可以指定以下值：

MQHC_UNASSOCIATED_HCONN

连接句柄不表示与任何特定队列管理器的连接。

使用此值时，必须通过显式调用 [MQDLTMH](#) 来删除消息句柄，以便释放分配给它的任何存储器；WebSphere MQ 从不隐式删除消息句柄。

必须至少有一个到创建消息句柄的线程上建立的队列管理器的有效连接，否则调用将失败并返回 MQRC_HCONN_ERROR。

在单个系统上具有多个安装的环境中，MQHC_UNASSOCIATED_HCONN 值仅限用于装入到进程中的第一个安装。如果将消息句柄提供给其他安装，那么将返回原因码 MQRC_HMSG_NOT_AVAILABLE。

在针对 CICS 应用程序的 z/OS 上，以及针对以兼容性方式运行的应用程序的 IBM i 上，可以省略 MQCONN 调用，并且可以为 *Hconn* 指定以下值：

MQHC_DEF_CONN

缺省连接句柄

CrtMsgHOpts

类型 :MQCMHO-输入

用于控制 MQCRTMH 操作的选项。请参阅 [MQCMHO](#) 以获取详细信息。

Hmsg

类型 :MQHMSG-输出

在输出时，将返回可用于设置，查询和删除消息句柄的属性的消息句柄。最初，消息句柄不包含任何属性。

消息句柄还具有关联的消息描述符。最初，这包含缺省值。可以使用 MQSETMP 和 MQINQMP 调用来设置和查询关联消息描述符字段的值。MQDLTMP 调用将消息描述符的字段重置回其缺省值。

如果将 *Hconn* 参数指定为值 MQHC_UNASSOCIATED_HCONN，那么可以将返回的消息句柄用于 MQGET，MQPUT 或 MQPUT1 调用以及处理单元中的任何连接，但一次只能由一个 WebSphere MQ 调

用使用。如果当第二个 WebSphere MQ 调用尝试使用同一消息句柄时正在使用该句柄，那么第二个 WebSphere MQ 调用将失败，原因码为 MQRC_MSG_HANDLE_IN_USE。

如果 *Hconn* 参数不是 MQHC_UNASSOCIATED_HCONN，那么只能在指定的连接上使用返回的消息句柄。

必须在使用此消息句柄的后续 MQI 调用上使用相同的 *Hconn* 参数值：

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

当针对消息句柄发出 MQDLTMH 调用时，或者当定义句柄作用域的处理单元终止时，返回的消息句柄将不再有效。如果在创建消息句柄时提供了特定连接，并且与队列管理器的连接不再有效 (例如，如果调用 MQDBC)，那么将隐式调用 MQDLTMH。

CompCode

类型：MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

如果 *CompCode* 为 MQCC_OK：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED：

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_CMHO_ERROR

(2461, X'099D') 创建消息句柄选项结构无效。

MQRC_CONNECTION_BROKEN

(2273, X'7D9') 与队列管理器的连接丢失。

MQRC_HANDLE_NOT_AVAILABLE

(2017, X'07E1') 没有更多可用的句柄。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_HMSG_ERROR

(2460, X'099C') 消息句柄指针无效。

MQRC_OPTIONS_ERROR

(2046, X'07FE') 选项无效或不一致。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

按如下所示声明参数：

```
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.
```

按如下所示声明参数：

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Options that control the action of MQCRTMH
01 CRTMSGOPTS.
   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);
```

按如下所示声明参数：

```
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, CRTMSGOPTS, HMSG, COMPCODE, REASON)
```

按如下所示声明参数：

```
HCONN      DS      F      Connection handle
CRTMSGOPTS 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-控制回调

MQCTL 调用对回调执行控制操作，并且为连接打开对象句柄。

语法

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Reason*)

参数

Hconn

类型 :MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNEX 调用返回了 *Hconn* 的值。

在 z/OS for CICS 应用程序上，以及 IBM i for applications in compatibility mode，可以省略 MQCONN 调用，并且可以为 *Hconn* 指定以下特殊值：

MQHC_DEF_HCONN

缺省连接句柄。

operation

类型 :MQLONG-输入

在为指定对象句柄定义的回调上正在处理的操作。必须指定下列选项中的一个 (且仅指定一个)：

MQOP_START

开始对指定连接句柄的所有已定义消息使用者函数使用消息。

回调在系统启动的线程上运行，这与任何应用程序线程不同。

此操作将控制提供给系统的连接句柄。可以由除使用者线程以外的线程发出的唯一 MQI 调用是：

- 具有操作 MQOP_STOP 的 MQCTL
- 具有操作 MQOP_SUSPEND 的 MQCTL
- MQDISC-在断开 HConn 连接之前使用操作 MQOP_STOP 执行 MQCTL。

如果在启动连接句柄时发出 WebSphere MQ API 调用，并且该调用并非源自消息使用者函数，那么将返回 MQRC_HCONN_ASYNC_ACTIVE。

如果消息使用者在 MQCBCT_START_CALL 期间停止连接，那么 MQCTL 调用将返回失败原因码 MQRC_CONNECTION_STOPPED。

这可以在使用者函数中发出。对于与回调例程相同的连接，其唯一目的是取消先前发出的 MQOP_STOP 操作。

此选项在以下环境中不受支持：CICS on z/OS，或者如果应用程序与非线程 WebSphere MQ 库绑定。

MQOP_START_WAIT

开始对指定连接句柄的所有已定义消息使用者函数使用消息。

在同一线程上运行的消息使用者和控制不会返回到 MQCTL 的调用者，直到：

- 通过使用 MQCTL MQOP_STOP 或 MQOP_SUSPEND 操作发布，或
- 已注销或暂挂所有使用者例程。

如果所有使用者都已注销或暂挂，那么将发出隐式 MQOP_STOP 操作。

此选项不能在回调例程中用于当前连接句柄或任何其他连接句柄。如果尝试调用，那么将返回 MQRC_environment_ERROR。

如果在 MQOP_START_WAIT 操作期间的任何时间没有已注册的非暂挂使用者，那么调用将失败，原因码为 MQRC_NO_CALLBACKS_ACTIVE。

如果在 MQOP_START_WAIT 操作期间暂挂了连接，那么 MQCTL 调用会返回警告原因码 MQRC_CONNECTION_SUSPENDED; 连接将保持 "已启动" 状态。

应用程序可以选择发出 MQOP_STOP 或 MQOP_RESUME。在此实例中，MQOP_RESUME 操作将阻塞。

此选项在单线程客户机中不受支持。

MQOP_STOP

停止使用消息，并在此选项完成之前等待所有使用者完成其操作。此操作将释放连接句柄。

如果从回调例程中发出，那么此选项在例程退出之前不会生效。在已读取的消息的使用者例程完成之后，以及在对回调例程进行停止调用 (如果请求) 之后，不再调用更多消息使用者例程。

如果在回调例程外部发出，那么直到已读消息的使用者例程完成，并且在对回调进行停止调用 (如果已请求) 之后，控制才会返回到调用者。但是，回调本身仍保持已注册状态。

此功能对预读消息没有影响。您必须确保使用者从回调函数中运行 MQCLOSE (MQCO_QUIESCE)，以确定是否有任何其他消息可供传递。

MQOP_SUSPEND

暂停使用消息。此操作将释放连接句柄。

这不会对应用程序的消息提前读取产生任何影响。如果打算长时间停止使用消息，请考虑关闭队列并在继续使用时重新打开该队列。

如果从回调例程中发出，那么直到例程退出后才会生效。在当前例程退出之后，将不再调用更多消息使用者例程。

如果在回调外部发出，那么直到当前使用者例程完成且不再调用时，控制才会返回到调用者。

MQOP_RESUME

恢复使用消息。

通常从主应用程序线程发出此选项，但也可以从回调例程中使用此选项来取消在同一例程中发出的较早暂挂请求。

如果使用 MQOP_RESUME 来恢复 MQOP_START_WAIT，那么操作将阻塞。

ControlOpts

类型:MQCTLO-输入

用于控制 MQCTL 的操作的选项

请参阅 [第 299 页的『MQCTLO-控制回调选项结构』](#) 以获取结构的详细信息。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 是 MQCC_FAILED:

MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') 无法装入数据转换服务模块。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') 缓冲区长度参数无效。

MQRC_CALLBACK_LINK_ERROR

(2487, X'9B7') 无法调用回调例程

MQRC_CALLBACK_NOT_已注册

(2448, X'990') 无法注销, 暂挂或恢复, 因为没有已注册的回调

MQRC_CALLBACK_ROUTINE_ERROR

(2486, X'9B6') 在 MQOP_REGISTER 调用上指定了 CallbackFunction 和 CallbackName。
或者指定了 CallbackFunction 或 CallbackName, 但与当前注册的回调函数不匹配。

MQRC_CALLBACK_TYPE_ERROR

(2483, X'9B3') CallBack 类型字段不正确。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CBD_ERROR

(2444, X'98C') 选项块不正确。

MQRC_CBD_OPTIONS_ERROR

(2484, X'9B4') MQCBD 选项字段不正确。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 未授权连接。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_CORREL_ID_ERROR

(2207, X'89F') 相关标识错误。

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') 请求的功能在当前环境中不可用。

MQRC_GET_禁止

(2016, X'7E0') 队列禁止获取。

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') 全局工作单元冲突。

MQRC_GMO_ERROR
(2186, X'88A') Get-message 选项结构无效。

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') 用于全局工作单元的句柄。

MQRC_HCONN_ERROR
(2018, X'7E2') 连接句柄无效。

MQRC_HOBJ_ERROR
(2019, X'7E3') 对象句柄无效。

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') 浏览规范不一致。

MQRC_INCONSISTENT_UOW
(2245, X'8C5') 工作单元规范不一致。

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') 游标下的消息对于检索无效。

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') 全局工作单元与本地工作单元冲突。

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') 匹配选项无效。

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B5') MaxMsg 长度字段不正确

MQRC_MD_ERROR
(2026, X'7EA') 消息描述符无效。

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') 在模块中找不到指定的函数入口点。

MQRC_MODULE_INVALID
(2496, X'9C0') 找到模块, 但其类型错误 (32 bit/64 位) 或不是有效的 dll。

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') 在搜索路径中找不到模块或无权装入。

MQRC_MSG_ID_ERROR
(2206, X'89E') 消息标识错误。

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') 消息序号无效。

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') 使用消息令牌无效。

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') 未打开队列以进行浏览。

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') 未打开队列以进行输入。

MQRC_OBJECT_CHANGED
(2041, X'7F9') 对象定义自打开以来已更改。

MQRC_OBJECT_DAMAGED
(2101, X'835') 对象已损坏。

MQRC_OPERATION_ERROR
(2488, X'9B8') API 调用上的操作码不正确

MQRC_OPTIONS_ERROR
(2046, X'7FE') 选项无效或不一致。

MQRC_PAGESET_ERROR
(2193, X'891') 访问页集数据集时出错。

MQRC_Q_DELETED
(2052, X'804') 队列已删除。

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') 队列具有错误的索引类型。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING

(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_SIGNAL_众数

(2069, X'815 ') 此句柄的信号未完成。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口程序禁止调用。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818 ') 同步点支持不可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932 ') 在全局工作单元中登记失败。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933 ') 不支持混合工作单元调用。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 工作单元不可供队列管理器使用。

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') MQGMO 中的等待时间间隔无效。

MQRC_不法_gmo_version

(2256, X'8D0') 提供的 MQGMO 版本不正确。

MQRC_不法_md_version

(2257, X'8D1') 提供的 MQMD 版本不正确。

有关这些代码的详细信息，请参阅原因码。

使用说明

1. 回调例程必须检查它们调用的所有服务的响应，如果例程检测到无法解析的条件，那么它必须发出 MQCB MQOP_DEREGISTER 命令以防止重复调用回调例程。
2. 在 z/OS 上，当操作为 MQOP_START 时：
 - 必须授权使用异步回调例程的程序使用 z/OS UNIX 系统服务 (USS)。
 - 使用异步回调例程的语言环境 (LE) 程序必须使用 LE 运行时选项 POSIX(ON)。
 - 使用异步回调例程的非 LE 程序不得使用 USS pthread_create 接口 (可调用服务 BPX1PTC)。
3. 在 IMS 适配器中不支持 MQCTL。

注：在 CICS 中，不支持 MQOP_START。请改为使用 MQOP_START_WAIT 函数调用。

C 调用

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.
```

PL/I 调用

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

按如下所示声明参数:

```
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-断开连接队列管理器

MQDISC 调用中断队列管理器与应用程序之间的连接，并且是 MQCONN 或 MQCONNX 调用的反向调用。

- 在 z/OS 上，所有使用异步消息消耗，事件处理或回调的应用程序，主控制线程必须在结束前发出 MQDISC 调用。有关更多详细信息，请参阅 [WebSphere MQ 消息的异步使用](#)。
- 在 z/OS 上，CICS 应用程序不需要发出此调用以与队列管理器断开连接，但可能需要发出此调用以结束连接标记的使用。
- 在 IBM i 上，以兼容性方式运行的应用程序不需要发出此调用。请参阅第 578 页的『MQCONN - 连接队列管理器』以获取更多信息。

语法

```
MQDISC (Hconn, CompCode, Reason)
```

参数

Hconn

类型:MQHCONN-输入/输出

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。

在 CICS 应用程序的 z/OS 上,以及在以兼容性方式运行的应用程序的 IBM i 上,可以省略 MQCONN 调用,并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

成功完成调用时,队列管理器会将 *Hconn* 设置为不是环境的有效句柄的值。此值为:

MQHC_UNUSABLE_HCONN

不可用的连接句柄。

在 z/OS 上, *Hconn* 设置为未定义的值。

CompCode

类型: MQLONG - 输出

完成代码;它是下列其中一个代码:

MQCC_OK

成功完成。

MQCC_WARNING

警告(部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_BACKED_OUT

(2003, X'7D3') 工作单元已回退。

MQRC_CONN_TAG_NOT_RELEASED

(2344, X'928') 未释放连接标记。

MQRC_OUTCOME_PENDING

(2124, X'84C') 落实操作的结果处于暂挂状态。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_DISC_LOAD_ERROR

(2138, X'85A') 无法装入适配器断开连接模块。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_INIT_ERROR

(2375, X'947') API 出口初始化失败。

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') API 出口终止失败。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_OUTCOME_MIXED

(2123, X'84B') 落实或回退操作的结果是混合的。

MQRC_PAGESET_ERROR

(2193, X'891') 访问页集数据集时出错。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

- 如果在连接仍打开该连接下的对象时发出 MQDISC 调用，那么队列管理器将关闭这些对象，并将关闭选项设置为 MQCO_NONE。
- 如果应用程序以工作单元中未落实的更改结束，那么这些更改的处置取决于应用程序的结束方式：
 - 如果应用程序在结束之前发出 MQDISC 调用：
 - 对于队列管理器协调的工作单元，队列管理器代表应用程序发出 MQCMIT 调用。如果可能，将落实工作单元，如果可能，将回退工作单元。
 - 对于外部协调的工作单元，工作单元的状态没有变化；但是，队列管理器通常指示工作单元必须在工作单元协调程序询问时落实。

在 z/OS 上，CICS，IMS (批处理 DL/1 程序除外) 和 RRS 应用程序与此类似。
 - 如果应用程序正常结束但未发出 MQDISC 调用，那么所执行的操作取决于环境：
 - 在 z/OS 上，除 MQ Java 或 MQ JMS 应用程序外，将发生注释 2a 中描述的操作。
 - 在所有其他情况下，会发生注释 2c 中描述的操作。

由于环境之间的差异，请确保要端口的应用程序在结束之前落实或回退工作单元。
 - 如果应用程序异常结束而不发出 MQDISC 调用，那么将回退工作单元。
- 在 z/OS 上，以下要点适用：
 - CICS 应用程序不必发出 MQDISC 调用来与队列管理器断开连接，因为 CICS 系统本身连接到队列管理器，而 MQDISC 调用对此连接没有影响。

- CICS, IMS (批处理 DL/1 程序除外) 和 RRS 应用程序使用由外部工作单元协调程序协调的工作单元。因此, MQDISC 调用不会影响发出调用时存在的工作单元 (如果有) 的状态。

但是, MQDISC 调用 确实 指示结束使用由应用程序发出的较早 MQCONN 调用与连接关联的连接标记 *ConnTag*。如果发出 MQDISC 调用时存在引用连接标记的活动工作单元, 那么调用将完成, 完成代码为 MQCC_WARNING, 原因码为 MQRC_CONN_TAG_NOT_RELEASED。直到外部工作单元协调程序已解析工作单元之后, 连接标记才可供复用。

4. 在 IBM i 上, 以兼容性方式运行的应用程序不必发出此调用; 请参阅 MQCONN 调用以获取更多详细信息。

注: 在 CICS 中, 不支持 MQOP_START。请改为使用 MQOP_START_WAIT 函数调用。

C 调用

```
MQDISC (&Hconn, &CompCode, &Reason);
```

按如下所示声明参数:

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

COBOL 调用

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.
```

PL/I 调用

```
call MQDISC (Hconn, CompCode, Reason);
```

按如下所示声明参数:

```
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 汇编程序调用

```
CALL MQDISC, (HCONN, COMPCODE, REASON)
```

按如下所示声明参数:

```
HCONN      DS F Connection handle
COMPCODE   DS F Completion code
REASON     DS F Reason code qualifying COMPCODE
```

Visual Basic 调用

```
MQDISC Hconn, CompCode, Reason
```

按如下所示声明参数:

```
Dim Hconn As Long 'Connection handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQDLTMH-删除消息句柄

MQDLTMH 调用将删除消息句柄, 并且是 MQCRTMH 调用的反向调用。

语法

```
MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason)
```

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。

该值必须与用于创建 *Hmsg* 参数中指定的消息句柄的连接句柄相匹配。

如果消息句柄是使用 MQHC_UNASSOCIATED_HCONN 创建的, 那么必须在删除消息句柄的线程上建立有效连接, 否则调用将失败, 并返回 MQRC_CONNECTION_BROKEN。

Hmsg

类型:MQHMSG-输入/输出

这是要删除的消息句柄。该值由先前的 MQCRTMH 调用返回。

成功完成调用时, 句柄将设置为环境的无效值。此值为:

MQHM_UNUSABLE_HMSG

不可用的消息句柄。

如果正在进行传递同一消息句柄的另一个 WebSphere MQ 调用, 那么无法删除该消息句柄。

DltMsgHOpts

类型:MQDMHO-输入

请参阅第 315 页的『MQDMHO-删除消息句柄选项』, 以了解详细信息。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') 与队列管理器的连接丢失。

MQRC_DMHO_ERROR

(2462, X'099E') 删除消息句柄选项结构无效。

MQRC_HMSG_ERROR

(2460, X'099C') 消息句柄指针无效。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') 消息句柄已在使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') 选项无效或不一致。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

C 调用

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

按如下所示声明参数：

```
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 */
```

COBOL 调用

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMGOPTS, COMPCODE, REASON.
```

按如下所示声明参数：

```
** Connection handle
01 HCONN PIC S9(9) BINARY.

** Options that control the action of MQDLTMH
01 DLTMGOPTS.
   COPY CMQDLMHOV.

** Completion code
01 COMPCODE PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

PL/I 调用

```
call MQDLTMH (Hconn, Hmsg, DltMsgH0pts, CompCode, Reason);
```

按如下所示声明参数:

```
decl Hconn      fixed bin(31); /* Connection handle */
decl Hmsg       fixed bin(63); /* Message handle */
decl DltMsgH0pts like MQDMHO; /* Options that control the action of MQDLTMH */
decl CompCode   fixed bin(31); /* Completion code */
decl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

高级汇编程序调用

```
CALL MQDLTMH, (HCONN, HMSG, DLTMMSGHOPTS, COMPCODE, REASON)
```

按如下所示声明参数:

| | | | |
|--------------|----------|---|--|
| HCONN | DS | F | Connection handle |
| HMSG | DS | D | Message handle |
| DLTMMSGHOPTS | CMQDMHOA | , | Options that control the action of MQDLTMH |
| COMPCODE | DS | F | Completion code |
| REASON | DS | F | Reason code qualifying COMPCODE |

MQDLTMP-删除消息属性

MQDLTMP 调用从消息句柄中删除属性，并且是 MQSETMP 调用的反向调用。

语法

```
MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason)
```

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。该值必须与用于创建 *Hmsg* 参数中指定的消息句柄的连接句柄相匹配。

如果消息句柄是使用 MQHC_UNASSOCIATED_HCONN 创建的，那么必须在删除消息句柄的线程上建立有效连接，否则调用将失败并导致 MQRC_CONNECTION_BROKEN。

Hmsg

类型:MQHMSG-输入

这是包含要删除的属性的消息句柄。该值由先前的 MQCRTMH 调用返回。

DltProp 选项

类型:MQDMPO-输入

请参阅 [MQDMPO](#) 数据类型以获取详细信息。

名称

类型:MQCHARV-输入

要删除的属性的名称。请参阅 [属性名](#) 以获取有关属性名的更多信息。

属性名称中不允许使用通配符。

CompCode

类型: MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_WARNING

警告（部分完成）。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

如果 *CompCode* 为 MQCC_OK：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING：

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') 属性不可用。

MQRC_RFH_FORMAT_ERROR

(2421, X'0975 ') 无法解析包含属性的 MQRFH2 文件夹。

如果 *CompCode* 为 MQCC_FAILED：

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'0852 ') 无法装入适配器服务模块。

MQRC_ASID_MISMATCH

(2157, X'086D') 主 ASID 和主 ASID 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') 与队列管理器的连接丢失。

MQRC_DMPO_ERROR

(2481, X'09B1') 删除消息属性选项结构无效。

MQRC_HMSG_ERROR

(2460, X'099C') 消息句柄无效。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') 消息句柄已在使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') 选项无效或不一致。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') 属性名无效。

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') 属性名编码字符集标识无效。

MQRC_UNEXPECTED_ERROR

(2195, X'0893 ') 发生意外错误。

有关这些代码的详细信息，请参阅：

- WebSphere MQ for z/OS 的 [原因码](#)
- 其他 WebSphere MQ 平台的 [API 原因码](#)

C 调用

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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 CMQCHRVA.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON  PIC S9(9) BINARY.
```

PL/I 调用

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

按如下所示声明参数:

```
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 */
```

高级汇编程序调用

```
CALL MQDLTMP, (HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON)
```

按如下所示声明参数:

| | | | |
|-------------|----------|---|--|
| 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-获取消息

MQGET 调用从使用 MQOPEN 调用打开的本地队列中检索消息。

语法

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

参数

Hconn

类型 :MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNEX 调用返回了 *Hconn* 的值。

在针对 CICS 应用程序的 z/OS 上, 以及针对以兼容性方式运行的应用程序的 IBM i 上, 可以省略 MQCONN 调用, 并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

Hobj

类型 :MQHOBJ-输入

此句柄表示要从中检索消息的队列。 *Hobj* 的值由先前的 MQOPEN 调用返回。必须已使用以下一个或多个选项打开队列 (请参阅第 642 页的『MQOPEN-打开对象』以获取详细信息):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

类型 :MQMD-输入/输出

此结构描述所需消息的属性以及检索的消息的属性。有关详细信息, 请参阅第 364 页的『MQMD - 消息描述符』。

如果 *BufferLength* 小于消息长度, 那么队列管理器将填充 *MsgDesc*, 无论是否在 *GetMsgOpts* 参数上指定 MQGMO_ACCEPT_TRUNCATED_MSG (请参阅 MQGMO-选项字段)。

如果应用程序提供 version-1 MQMD, 那么返回的消息具有前缀为应用程序消息数据的 MQMDE, 但仅如果 MQMDE 中的一个或多个字段具有非缺省值。如果 MQMDE 中的所有字段都具有缺省值, 那么将省略 MQMDE。MQMD 中的格式字段中的 MQFMT_MD_EXTENSION 格式名称指示存在 MQMDE。

如果在 *MsgHandle* 字段中提供了有效的消息句柄, 那么应用程序不需要提供 MQMD 结构。如果此字段中未提供任何内容, 那么将从与消息句柄关联的描述符中获取消息的描述符。

如果应用程序提供消息句柄而不是 MQMD 结构, 并指定 MQGMO_PROPERTIES_FORCE_MQRFH2, 那么调用将失败, 原因码为 MQRC_MD_ERROR。如果应用程序未提供 MQMD 结构并指定 MQGMO_PROPERTIES_AS_Q_DEF, 并且 *PropertyControl* 队列属性为 MQPROP_FORCE_MQRFH2, 那么调用也将失败, 原因码为 MQRC_MD_ERROR。

如果指定了匹配选项并且正在使用与消息句柄相关联的消息描述符, 那么用于匹配的输入字段来自消息句柄。

GetMsg 选项

类型 :MQGMO-输入/输出

请参阅第 324 页的『MQGMO-Get-消息选项』, 以了解详细信息。

BufferLength

类型 :MQLONG-输入

这是 *Buffer* 区域的长度 (以字节计)。为没有数据的消息指定零, 或者如果要从队列中除去消息并废弃数据 (在这种情况下, 必须指定 MQGMO_ACCEPT_TRUNCATED_MSG)。

注: 可从队列中读取的最长消息的长度由 *MaxMsgLength* 队列属性提供; 请参阅第 731 页的『队列的属性』。

buffer

类型: MQBYTEExBuffer 长度-输出

这是用于包含消息数据的区域。将缓冲区与消息中数据的性质相应的边界对齐。4 字节对齐适用于大多数消息 (包括包含 IBM WebSphere MQ 头结构的消息), 但某些消息可能需要更严格的对齐。例如, 包含 64 位二进制整数的消息可能需要 8 字节对齐。

如果 *BufferLength* 小于消息长度, 那么会将尽可能多的消息移动到 *Buffer* 中; 如果在 *GetMsgOpts* 参数上指定了 MQGMO_ACCEPT_TRUNCATED_MSG (请参阅 [MQGMO-选项字段](#) 以获取更多信息), 那么会发生此情况。

Buffer 中数据的字符集和编码由 *MsgDesc* 参数中返回的 *CodedCharSetId* 和 *Encoding* 字段提供。如果这些值与接收方所需的值不同, 那么接收方必须将应用程序消息数据转换为所需的字符集和编码。可以使用 MQGMO_CONVERT 选项 (必要时带有用户编写的出口) 来转换消息数据; 请参阅 [第 324 页的『MQGMO-Get-消息选项』](#) 以获取此选项的详细信息。

注: MQGET 调用上的所有其他参数都采用本地队列管理器的字符集和编码 (由 *CodedCharSetId* queue-manager 属性和 MQENC_NATIVE 提供)。

如果调用失败, 那么缓冲区的内容可能仍已更改。

在 C 编程语言中, 参数被声明为一个指向 void 的指针: 任何类型数据的地址都可以被指定为参数。

如果 *BufferLength* 参数为零, 那么不会引用 *Buffer*; 在此情况下, 以 C 或 System/390 汇编程序编写的程序传递的参数地址可以为空。

DataLength

类型: MQLONG - 输出

这是应用程序数据在消息中的长度 (以字节计)。如果此值大于 *BufferLength*, 那么在 *Buffer* 参数中仅返回 *BufferLength* 字节 (即, 消息被截断)。如果值为零, 那么消息不包含应用程序数据。

如果 *BufferLength* 小于消息长度, 那么 *DataLength* 仍由队列管理器完成, 无论是否在 *GetMsgOpts* 参数上指定 MQGMO_ACCEPT_TRUNCATED_MSG (请参阅 [MQGMO-选项字段](#) 以获取更多信息)。这允许应用程序确定容纳消息数据所需的缓冲区大小, 然后使用适当大小的缓冲区重新发出调用。

但是, 如果指定了 MQGMO_CONVERT 选项, 并且转换后的消息数据太长, 无法放入 *Buffer* 中, 那么针对 *DataLength* 返回的值为:

- 队列管理器定义的格式的未转换数据的长度。

在这种情况下, 如果数据的性质导致其在转换期间扩展, 那么应用程序必须为 *DataLength* 分配大于队列管理器返回的值的缓冲区。

- 数据转换出口针对应用程序定义的格式返回的值。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

列出的原因码是队列管理器可以针对 *Reason* 参数返回的原因码。如果应用程序指定了 MQGMO_CONVERT 选项, 并且调用了用户编写的出口来转换部分或所有消息数据, 那么该出口将决定为 *Reason* 参数返回的值。因此, 记录的值以外的值是可能的。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') 转换的数据对于缓冲区太大。

MQRC_CONVERTED_STRING_TOO_BIG

(2190, X'88E') 转换的字符串对于字段太大。

MQRC_DBCS_ERROR

(2150, X'866') DBCS 字符串无效。

MQRC_FORMAT_ERROR

(2110, X'83E') 消息格式无效。

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') 消息组未完成。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 逻辑消息未完成。

MQRC_INCONSISTENT_CCSDS

(2243, X'8C3') 消息段具有不同的 CCSID。

MQRC_INCONSISTENT_ENCODINGS

(2244, X'8C4') 消息段具有不同的编码。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 工作单元规范不一致。

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') 消息令牌的使用无效。

MQRC_NO_MSG_LOCKED

(2209, X'8A1') 未锁定任何消息。

MQRC_NOT_汇率

(2119, X'847') 未转换消息数据。

MQRC_OPTIONS_CHANGED

(nnnn, X'xxx') 已更改需要一致的选项。

MQRC_PARTIALLY_汇率

(2272, X'8E0') 消息数据已部分转换。

MQRC_SIGNAL_REQUEST_接受

(2070, X'816') 未返回消息 (但接受信号请求)。

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861') 源缓冲区参数无效。

MQRC_SOURCE_CCSDS_ERROR

(2111, X'83F') 源编码字符集标识无效。

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841') 无法识别消息中的压缩十进制编码。

MQRC_SOURCE_FLOAT_ENC_ERROR

(2114, X'842') 无法识别消息中的浮点编码。

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') 无法识别源整数编码。

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') 源长度参数无效。

MQRC_TARGET_BUFFER_ERROR

(2146, X'862') 目标缓冲区参数无效。

MQRC_TARGET_CCSDS_ERROR

(2115, X'843') 目标编码字符集标识无效。

MQRC_TARGET_DECIMAL_ENC_ERROR

(2117, X'845') 接收器指定的压缩十进制编码无法识别。

MQRC_TARGET_FLOAT_ENC_ERROR

(2118, X'846') 接收器指定的浮点编码无法识别。

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') 无法识别目标整数编码。

MQRC_TRUNCATED_MSG_RECEIVED

(2079, X'81F') 已返回截断的消息 (处理已完成)。

MQRC_TRUNCATED_MSG_FAILED

(2080, X'820') 已返回截断的消息 (处理未完成)。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') 无法装入数据转换服务模块。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BACKED_OUT

(2003, X'7D3') 工作单元已回退。

MQRC_BUFFER_ERROR

(2004, X'7D4') 缓冲区参数无效。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CF_STRUC_FAILED

(2373, X'945') 耦合设施结构失败。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') 正在使用耦合设施结构 list-header。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 未授权连接。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_CORREL_ID_ERROR

(2207, X'89F') 相关标识错误。

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') 数据长度参数无效。

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') Db2 子系统不可用。

MQRC_GET_禁止
(2016, X'7E0') 队列禁止获取。

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') 全局工作单元冲突。

MQRC_GMO_ERROR
(2186, X'88A') Get-message 选项结构无效。

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') 用于全局工作单元的句柄。

MQRC_HCONN_ERROR
(2018, X'7E2') 连接句柄无效。

MQRC_HOBJ_ERROR
(2019, X'7E3') 对象句柄无效。

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') 浏览规范不一致。

MQRC_INCONSISTENT_UOW
(2245, X'8C5') 工作单元规范不一致。

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') 游标下的消息对于检索无效。

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') 全局工作单元与本地工作单元冲突。

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') 匹配选项无效。

MQRC_MD_ERROR
(2026, X'7EA') 消息描述符无效。

MQRC_MSG_ID_ERROR
(2206, X'89E') 消息标识错误。

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') 消息序号无效。

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') 使用消息令牌无效。

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') 无消息可用。

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') 浏览光标未定位在消息上。

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') 未打开队列以进行浏览。

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') 未打开队列以进行输入。

MQRC_OBJECT_CHANGED
(2041, X'7F9') 对象定义自打开以来已更改。

MQRC_OBJECT_DAMAGED
(2101, X'835') 对象已损坏。

MQRC_OPTIONS_ERROR
(2046, X'7FE') 选项无效或不一致。

MQRC_PAGESET_ERROR
(2193, X'891') 访问页集数据集时出错。

MQRC_Q_DELETED

(2052, X'804') 队列已删除。

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') 队列具有错误的索引类型。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING

(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_SECOND_MARK_NOT_ALLOWED

(2062, X'80E') 已标记消息。

MQRC_SIGNAL_众数

(2069, X'815') 此句柄的信号未完成。

MQRC_SIGNAL1_ERROR

(2099, X'833') 信号字段无效。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部存储介质已满。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口程序禁止调用。

MQRC_SYNCPOINT_LIMIT_已达到

(2024, X'7E8') 无法在当前工作单元中处理更多消息。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') 同步点支持不可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') 在全局工作单元中登记失败。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') 不支持混合工作单元调用。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 工作单元不可供队列管理器使用。

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') MQGMO 中的等待时间间隔无效。

MQRC_不法_gmo_version

(2256, X'8D0') 提供的 MQGMO 版本不正确。

MQRC_不法_md_version

(2257, X'8D1') 提供的 MQMD 版本不正确。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

1. 通常会从队列中删除检索到的消息。此删除可作为 MQGET 调用本身的一部分进行，也可作为同步点的一部分进行。

浏览选项包括 :MQGMO_BROWSE_FIRST , MQGMO_BROWSE_NEXT 和 MQGMO_BROWSE_MSG_UNDER_CURSOR。

2. 如果使用其中一个浏览选项指定了 MQGMO_LOCK 选项, 那么将锁定已浏览的消息, 以使其仅对此句柄可见。

如果指定了 MQGMO_UNLOCK 选项, 那么将解锁先前锁定的消息。在此情况下, 不会检索任何消息, 并且不会检查或变更 *MsgDesc*, *BufferLength*, *Buffer* 和 *DataLength* 参数。

3. 对于发出 MQGET 调用的应用程序, 如果应用程序异常终止或在处理调用时断开连接, 那么检索到的消息可能会丢失。产生此问题的原因是, 在代表应用程序发出 MQGET 调用的队列管理器所在的平台上运行的代理程序无法检测应用程序丢失的情况, 直到代理程序将消息返回到应用程序, 之后该消息已从队列中除去。持久消息和非持久消息都可能发生此问题。

要消除以这种方式丢失消息的风险, 请始终在工作单元中检索消息。即, 通过在 MQGET 调用上指定 MQGMO_SYNCPOINT 选项, 并在消息处理完成时使用 MQCMIT 或 MQBACK 调用来落实或回退工作单元。如果指定了 MQGMO_SYNCPOINT, 并且客户机异常终止或已断开连接, 那么代理将回退队列管理器上的工作单元, 并在队列上恢复消息。有关同步点的更多信息, 请参阅 [WebSphere MQ 应用程序中的同步点注意事项](#)。

对于 IBM WebSphere MQ 客户机以及与队列管理器在同一平台上运行的应用程序, 可能会出现此情况。

4. 如果应用程序将消息序列放在特定在单个工作单元中排队, 然后成功落实该工作单元, 这些消息可供检索, 如下所示:

- 如果队列是 非共享 队列 (即本地队列), 那么工作单元中的所有消息将同时可用。
- 如果队列是 共享 队列, 那么工作单元中的消息将按其放入的顺序变为可用, 但不能同时放入所有消息。当系统大量负载时, 可以成功检索工作单元中的第一条消息, 但对于工作单元中的第二条消息或后续消息的 MQGET 调用可能会失败, 并返回 MQRC_NO_MSG_AVAILABLE。如果发生此问题, 那么应用程序必须稍等片刻, 然后重试该操作。

5. 如果应用程序在不使用消息组的情况下将消息序列放在同一队列上, 如果满足特定条件, 那么将保留这些消息的顺序。请参阅 [MQPUT 使用说明](#) 以获取详细信息。如果满足条件, 那么将按照发送消息的顺序向接收应用程序提供消息, 前提是:

- 只有一个接收方从队列中获取消息。

如果有两个或更多应用程序从队列中获取消息, 那么它们必须与发送方同意用于标识属于序列的消息的机制。例如, 发送方可以将序列中的消息中的所有 *CorrelId* 字段设置为该消息序列所独有的值。

- 接收方不会故意更改检索顺序, 例如通过指定特定 *MsgId* 或 *CorrelId*。

如果发送应用程序将消息作为消息组放置, 那么如果接收应用程序在 MQGET 调用上指定了 MQGMO_LOGICAL_ORDER 选项, 那么将以正确顺序向接收应用程序显示消息。有关消息组的更多信息, 请参阅:

- [MQMD- MsgFlags 字段](#)
- [MQPMO_LOGICAL_ORDER](#)
- [MQGMO_LOGICAL_ORDER](#)

如果用户正在同步点下的组中获取消息, 那么他们必须确保在尝试完成事务之前处理完整组。

6. 应用程序必须在 *MsgDesc* 参数的 *Feedback* 字段中测试反馈代码 MQFB_QUIT, 如果它们找到此值, 那么结束。请参阅 [MQMD-反馈字段](#) 以获取更多信息。
7. 如果使用 MQOO_SAVE_ALL_CONTEXT 选项打开了由 *Hobj* 标识的队列, 并且 MQGET 调用的完成代码为 MQCC_OK 或 MQCC_WARNING, 那么与队列句柄 *Hobj* 关联的上下文将设置为已检索的消息的上下文 (除非设置了 MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT 或 MQGMO_BROWSE_MSG_UNDER_CURSOR 选项, 在这种情况下, 将上下文标记为不可用)。

您可以通过指定 MQPMO_PASS_IDENTITY_CONTEXT 或 MQPMO_PASS_ALL_CONTEXT 选项, 在后续 MQPUT 或 MQPUT1 调用上使用保存的上下文。这使接收到的消息的上下文能够全部或部分传输到另一个消息 (例如, 当消息转发到另一个队列时)。有关消息上下文的更多信息, 请参阅 [消息上下文](#)。

8. 如果在 *GetMsgOpts* 参数中包含 MQGMO_CONVERT 选项, 那么在将数据放入 *Buffer* 参数之前, 会将应用程序消息数据转换为接收应用程序请求的表示:

- 消息中的控制信息中的 *Format* 字段标识应用程序数据的结构，而消息中的控制信息中的 *CodedCharSetId* 和 *Encoding* 字段指定其字符集标识和编码。
- 发出 MQGET 调用的应用程序在 *MsgDesc* 参数的 *CodedCharSetId* 和 *Encoding* 字段中指定要将应用程序消息数据转换为的字符集标识和编码。

当需要转换消息数据时，将由队列管理器本身或用户编写的出口执行转换，具体取决于消息中的控制信息中 *Format* 字段的值：

- 以下格式名称是由队列管理器转换的格式；这些格式称为 "内置" 格式：
 - MQFMT_ADMIN
 - MQFMT_CICS (仅限 z/OS)
 - MQFMT_COMMAND_1
 - MQFMT_COMMAND_2
 - MQFMT_DEAD_LETTER_HEADER
 - MQFMT_DIST_HEADER
 - MQFMT_EVENT V 1
 - MQFMT_EVENT V 2 (仅限 z/OS)
 - MQFMT_IMS
 - MQFMT_IMS_VAR_STRING
 - MQFMT_MD_EXTENSION
 - MQFMT_PCF
 - MQFMT_REF_MSG_HEADER
 - MQFMT_RF_HEADER
 - MQFMT_RF_HEADER_2
 - MQFMT_STRING
 - MQFMT_TRIGGER
 - MQFMT_WORK_INFO_HEADER (仅限 z/OS)
 - MQFMT_XMIT_Q_HEADER
- 格式名 MQFMT_NONE 是一个特殊值，指示未定义消息中数据的性质。因此，从队列中检索消息时，队列管理器不会尝试转换。

注：如果在 MQGET 调用上为格式名为 MQFMT_NONE 的消息指定了 MQGMO_CONVERT，并且该消息的字符集或编码与 *MsgDesc* 参数中指定的字符集或编码不同，那么将在 *Buffer* 参数中返回该消息（假定没有其他错误），但该调用将完成，完成代码为 MQCC_WARNING，原因码为 MQRC_FORMAT_ERROR。

当消息数据的性质意味着它不需要转换时，或者当发送和接收应用程序已在它们之间商定用于发送消息数据的格式时，可以使用 MQFMT_NONE。

- 所有其他格式名称将消息传递到用户编写的出口以进行转换。除特定于环境的添加外，出口具有与格式相同的名称。用户指定的格式名称不得以字母 WebSphere MQ 开头。

有关数据转换出口的详细信息，请参阅第 791 页的『数据转换出口』。

可以在任何受支持的字符集和编码之间转换消息中的用户数据。但是，请注意，如果消息包含一个或多个 WebSphere MQ 头结构，那么不能将消息转换为对于队列名称中有效的任何字符具有双字节或多字节字符的字符集。如果尝试执行此操作，那么会生成原因码 MQRC_SOURCE_CCSDID_ERROR 或 MQRC_TARGET_CCSDID_ERROR，并且将返回未转换的消息。Unicode 字符集 UCS-2 是此类字符集的示例。

从 MQGET 返回时，以下原因码指示消息已成功转换：

- MQRC_NONE

以下原因码指示消息可能已成功转换；应用程序必须检查 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段以找出：

- MQRC_TRUNCATED_MSG_RECEIVED

所有其他原因码都指示未转换消息。

注: 此原因码的解释 仅 适用于用户编写的出口执行的转换，前提是该出口符合 [第 791 页](#) 的『数据转换出口』中描述的处理准则。

9. 使用面向对象的接口获取消息时，可以选择不指定缓冲区来保存 MQGET 调用的消息数据。但是，在先前版本的 WebSphere MQ 中，MQGET 可能会失败，原因码为 MQRC_CONVERTED_MSG_TO_BIG，即使未指定缓冲区也是如此。在 WebSphere MQ Version 7 中，当您在不限接收消息缓冲区大小的情况下使用面向对象的应用程序来获取消息时，应用程序不会因 MQRC_CONVERTED_MSG_TOO_BIG 而失败，并接收转换后的消息。对于以下环境，情况如此：

- .NET，包括完全受管的应用程序
- C++
- Java (WebSphere MQ Java 类)

注: 对于所有客户机，如果 *sharingConversations* 的值为零，那么通道将像 WebSphere MQ V 7.0 之前那样运行，并且消息处理将还原为 V 6 行为。在这种情况下，如果缓冲区太小而无法接收转换后的消息，那么将返回未转换的消息，原因码为 MQRC_CONVERTED_MSG_TOO_BIG。有关 *sharingConversations* 的更多信息，请参阅 [在客户机应用程序中使用共享对话](#)。

10. 对于内置格式，当指定 MQGMO_CONVERT 选项时，队列管理器可以执行消息中字符串的缺省转换。缺省转换允许队列管理器在转换字符串数据时使用安装指定的缺省字符集，该字符集与实际字符集近似。因此，MQGET 调用可以使用完成代码 MQCC_OK 成功，而不是使用 MQCC_WARNING 和原因码 MQRC_SOURCE_CCSID_ERROR 或 MQRC_TARGET_CCSID_ERROR 完成。

注: 使用近似字符集来转换字符串数据的结果是某些字符可能转换不正确。要避免这种情况，请在字符串中使用对实际字符集和缺省字符集都通用的字符。

缺省转换适用于应用程序消息数据以及 MQMD 和 MQMDE 结构中的字符字段：

- 仅当 所有 满足以下条件时，才会发生应用程序消息数据的缺省转换：
 - 应用程序指定 MQGMO_CONVERT。
 - 该消息包含必须从不支持的字符集转换或转换为不支持的字符集的数据。
 - 安装或重新启动队列管理器时，已启用缺省转换。
- 如果为队列管理器启用了缺省转换，那么将根据需要对 MQMD 和 MQMDE 结构中的字符字段进行缺省转换。即使 MQGET 调用上的应用程序未指定 MQGMO_CONVERT 选项，也会执行转换。

11. 对于 Visual Basic 编程语言，以下要点适用：

- 如果 *Buffer* 参数的大小小于 *BufferLength* 参数指定的长度，那么调用将失败，原因码为 MQRC_STORAGE_NOT_AVAILABLE。
- *Buffer* 参数声明为 *String* 类型。如果要从队列中检索的数据的类型不是 *String*，请使用 MQGETAny 调用代替 MQGET。

MQGETAny 调用具有与 MQGET 调用相同的参数，但 *Buffer* 参数声明为 *Any* 类型，允许检索任何类型的数据。但是，这意味着无法检查 *Buffer* 以确保其大小至少为 *BufferLength* 字节。

12. 当启用预读时，并非所有 MQGET 选项都受支持。下表指示允许哪些选项以及是否可以在 MQGET 调用之间更改这些选项。

| | 在启用预读时允许，并且可以在 MQGET 调用之间进行更改 | 在启用预读但无法在 MQGET 调用之间更改时允许 ^a | 启用预读时不允许的 MQGET 选项 ^b |
|------------|---|--|---------------------------------|
| MQGET MD 值 | MsgId ^c CorrelId ^c | 编码 CodedCharSetId | |

| 表 566: 启用预读时允许的 MQGET 选项 (继续) | | | |
|-------------------------------|---|--|--|
| | 在启用预读时允许, 并且可以在 MQGET 调用之间进行更改 | 在启用预读但无法在 MQGET 调用之间更改时允许 ^a | 启用预读时不允许的 MQGET 选项 ^b |
| MQGET MQGMO 选项 | 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 备份 (_BACKOUT) MQGMO_MSG_UNDER_CURSOR ^d MQGMO_LOCK MQGMO_UNLOCK |
| MQGMO 值 | | MsgHandle | |

- a. 如果在 MQGET 调用之间更改了这些选项, 那么将返回 MQRC_OPTIONS_CHANGED 原因码。
 - b. 如果在第一个 MQGET 调用上指定这些选项, 那么将禁用预读。如果在后续 MQGET 调用上指定这些选项, 那么将返回原因码 MQRC_OPTIONS_ERROR。
 - c. 客户机应用程序需要意识到, 如果 MsgId 和 CorrelId 值在 MQGET 调用之间发生更改, 那么具有先前值的消息可能已发送至客户机, 并保留在客户机预读缓冲区中, 直至被使用 (或自动清除) 为止。
 - d. 第一个 MQGET 调用确定在启用了预读时是否要从队列中浏览或获取消息。如果应用程序尝试同时执行浏览与获取操作, 将返回原因码 MQRC_OPTIONS_CHANGED。
 - e. MQGMO_MSG_UNDER_CURSOR 不能与预读配合使用。在启用了预读时, 可浏览或获取消息, 但是不能同时执行这两个操作。
13. 仅当将未落实的消息放入与 get 相同的本地工作单元时, 应用程序才能以破坏性方式获取未落实的消息。应用程序无法以非破坏性方式获取未落实的消息。
 14. 可以在工作单元中检索浏览光标下的消息。无法以此方式检索未落实的消息。

C 调用

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

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
                 BUFFER, DATALENGTH, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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 CMQGMV.
** 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.

```

PL/I 调用

```

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

```

按如下所示声明参数:

```

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 */

```

高级汇编程序调用

```

CALL MQGET,(HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,
BUFFER,DATALENGTH,COMPCODE,REASON)

```

按如下所示声明参数:

| | | | |
|--------------|---------|-------|--|
| 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 |

Visual Basic 调用

```

MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason

```

按如下所示声明参数:

```

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-查询对象属性

MQINQ 调用返回整数数组和一组包含对象属性的字符串。

以下类型的对象有效:

- 队列管理器
- 队列
- 名称列表
- 进程定义

语法

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Reason*)

参数

Hconn

类型: MQHCONN -输入

此句柄表示与队列管理器的连接。 *Hconn* 的值由先前的 MQCONN 或 MQCONNX 调用返回。

在 z/OS for CICS 应用程序和 IBM i 上, 对于以兼容性方式运行的应用程序, 可以省略 MQCONN 调用, 并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

Hobj

类型: MQHOBJ -输入

此句柄表示具有必需属性的对象 (任何类型)。 指定了 MQOO_INQUIRE 选项的先前 MQOPEN 调用必须返回句柄。

SelectorCount

类型: MQLONG -输入

这是 *Selectors* 数组中提供的选择器的计数。 这是要返回的属性数。 零是有效值。 允许的最大数目为 256。

选择器

类型: MQLONG × *SelectorCount* -输入

这是 *SelectorCount* 属性选择器的数组; 每个选择器都标识一个具有必需值的属性 (整数或字符)。

每个选择器都必须对 *Hobj* 表示的对象类型有效, 否则调用将失败, 并返回完成代码 MQCC_FAILED 和原因码 MQRC_SELECTOR_ERROR。

在队列的特殊情况下:

- 如果选择器对于任何类型的队列都无效, 那么调用将失败, 完成代码为 MQCC_FAILED, 原因码为 MQRC_SELECTOR_ERROR。
- 如果选择器仅应用于对象类型以外的类型的队列, 那么调用将成功, 完成代码为 MQCC_WARNING, 原因码为 MQRC_SELECTOR_NOT_FOR_TYPE。
- 如果要查询的队列是集群队列, 那么有效的选择器取决于解析队列的方式; 请参阅 [第 631 页的『使用说明』](#) 以获取更多详细信息。

您可以按任何顺序指定选择器。 对应于整数属性选择器 (MQIA_* 选择器) 的属性值将以这些选择器在 *Selectors* 中出现的相同顺序在 *IntAttrs* 中返回。 与字符属性选择器 (MQCA_* 选择器) 对应的属性

值在 *CharAttrs* 中按这些选择器的出现顺序返回。MQIA_* 选择器可以与 MQCA_* 选择器交错; 只有每种类型中的相对顺序很重要。

注:

1. 在两个不同的范围内分配整数和字符属性选择器; MQIA_* 选择器位于范围 MQIA_FIRST 到 MQIA_LAST 内, MQCA_* 选择器位于范围 MQCA_FIRST 到 MQCA_LAST 内。
对于每个范围, 常量 MQIA_LAST_USED 和 MQCA_LAST_USED 定义队列管理器接受的最大值。
2. 如果首先出现所有 MQIA_* 选择器, 那么可以使用相同的元素编号来寻址 *Selectors* 和 *IntAttrs* 数组中的相应元素。
3. 如果 *SelectorCount* 参数为零, 那么不会引用 *Selectors*。在这种情况下, 以 C 或 S/390 汇编程序编写的程序传递的参数地址可能为空。

下表列出了可查询的属性。对于 MQCA_* 选择器, 在括号中提供了用于定义 *CharAttrs* 中生成的字符串的长度 (以字节为单位) 的常量。

下面的表按对象按字母顺序列出选择器, 如下所示:

- 队列的 [第 620 页的表 567 MQINQ](#) 属性选择器
- 名称列表的 [第 622 页的表 568 MQINQ](#) 属性选择器
- 进程定义的 [第 623 页的表 569 MQINQ](#) 属性选择器
- 队列管理器的 [第 623 页的表 570 MQINQ](#) 属性选择器

所有 IBM WebSphere MQ 平台上都支持所有选择器, 但 **注释** 列中指示的选择器除外, 如下所示:

非 z/OS

在除 z/OS 之外的所有平台上受支持

z/OS

仅在 z/OS 上受支持

| 选择器 | 字段长度 | 描述 | 注 |
|-------------------------|-------------------------|------------------------|--------|
| MQCA_ALTERATION_DATE | MQ_DATE_LENGTH | 最近一次更改日期 | |
| MQCA_ALTERATION_TIME | MQ_TIME_LENGTH | 最近一次更改的时间 | |
| MQCA_BACKOUT_REQ_Q_NAME | MQ_Q_NAME_LENGTH | 过多的回退重新排队名称 | |
| MQCA_BASE_Q_NAME | MQ_Q_NAME_LENGTH | 别名解析为的队列的名称 | |
| MQCA_CF_STRUC_NAME | MQ_CF_STRUC_NAME_LENGTH | 耦合设施结构名称 | z/OS |
| MQCA_CLUS_CHL_NAME | MQ_CHANNEL_NAME_LENGTH | 将此队列用作传输队列的集群发送方通道的名称。 | 非 z/OS |
| MQCA_CLUSTER_NAME | MQ_CLUSTER_NAME_LENGTH | 集群名称 | |
| MQCA_CLUSTER_NAMELIST | MQ_NAMELIST_NAME_LENGTH | 集群名称列表 | |
| MQCA_CREATION_DATE | MQ_CREATION_DATE_LENGTH | 队列创建日期 | |
| MQCA_CREATION_TIME | MQ_CREATION_TIME_LENGTH | 队列创建时间 | |
| MQCA_INITIATION_Q_NAME | MQ_Q_NAME_LENGTH | 启动队列名称 | |
| MQCA_PROCESS_NAME | MQ_PROCESS_NAME_LENGTH | 进程定义的名称 | |

| 表 567: 队列的 MQINQ 属性选择器 (继续) | | | |
|-----------------------------|-------------------------|-----------------------|--------|
| 选择器 | 字段长度 | 描述 | 注 |
| MQCA_Q_DESC | MQ_Q_DESC_LENGTH | 队列描述 | |
| MQCA_Q_NAME | MQ_Q_NAME_LENGTH | 队列名称 | |
| MQCA_REMOTE_Q_MGR_NAME | MQ_Q_MGR_NAME_LENGTH | 远程队列管理器的名称 | |
| MQCA_REMOTE_Q_NAME | MQ_Q_NAME_LENGTH | 远程队列管理器上已知的远程队列的名称 | |
| MQCA_STORAGE_CLASS | MQ_STORAGE_CLASS_LENGTH | 存储类的名称 | z/OS |
| MQCA_TRIGGER_DATA | MQ_TRIGGER_DATA_LENGTH | 触发器数据 | |
| MQCA_XMIT_Q_NAME | MQ_Q_NAME_LENGTH | 传输队列的名称 | |
| MQIA_ACCOUNTING_Q | MQLONG | 控制队列的记帐数据收集 | 非 z/OS |
| MQIA_BACKOUT_THRESHOLD | MQLONG | 回退阈值 | |
| MQIA_CLWL_Q_PRIORITY | MQLONG | 队列的优先级 | |
| MQIA_CLWL_Q_RANK | MQLONG | 队列的排名 | |
| MQIA_CLWL_USEQ | MQLONG | 使用远程队列 | |
| MQIA_CURRENT_Q_DEPTH | MQLONG | 队列中的消息数 | |
| MQIA_DEF_BIND | MQLONG | 缺省绑定 | |
| MQIA_DEF_INPUT_OPEN_OPTION | MQLONG | 缺省 open-for-input 选项 | |
| MQIA_DEF_PERSISTENCE | MQLONG | 缺省消息持久性 | |
| MQIA_DEF_PRIORITY | MQLONG | 缺省消息优先级 | |
| MQIA_DEFINITION_TYPE | MQLONG | 队列定义类型 | |
| MQIA_DIST_LISTS | MQLONG | 分发列表支持 | 非 z/OS |
| MQIA_HARDEN_GET_BACKOUT | MQLONG | 是否硬化回退计数 | |
| MQIA_INDEX_TYPE | MQLONG | 为队列维护的索引类型 | z/OS |
| MQIA_INHIBIT_GET | MQLONG | 是否允许获取操作 | |
| MQIA_INHIBIT_PUT | MQLONG | 是否允许 put 操作 | |
| MQIA_MAX_MSG_LENGTH | MQLONG | 最大消息长度 | |
| MQIA_MAX_Q_DEPTH | MQLONG | 队列上允许的最大消息数 | |
| MQIA_MSG_DELIVERY_SEQUENCE | MQLONG | 消息优先级是否相关 | |
| MQIA_NPM_CLASS | MQLONG | 非持久消息的可靠性级别 | |
| MQIA_OPEN_INPUT_COUNT | MQLONG | 打开队列以进行输入的 MQOPEN 调用数 | |
| MQIA_OPEN_OUTPUT_COUNT | MQLONG | 打开队列以进行输出的 MQOPEN 调用数 | |
| MQIA_PROPERTY_CONTROL | MQLONG | 属性控制属性 | |

| 选择器 | 字段长度 | 描述 | 注 |
|-------------------------------|--------|-----------------|--------|
| MQIA_Q_DEPTH_HIGH_EVENT | MQLONG | 队列深度高事件的控制属性 | 非 z/OS |
| MQIA_Q_DEPTH_HIGH_LIMIT | MQLONG | 队列深度的上限 | 非 z/OS |
| MQIA_Q_DEPTH_LOW_EVENT | MQLONG | 队列深度低事件的控制属性 | 非 z/OS |
| MQIA_Q_DEPTH_LOW_LIMIT | MQLONG | 队列深度的下限 | 非 z/OS |
| MQIA_Q_DEPTH_MAX_EVENT | MQLONG | 队列深度最大事件数的控制属性 | 非 z/OS |
| MQIA_Q_SERVICE_INTERVAL | MQLONG | 队列服务时间间隔限制 | 非 z/OS |
| MQIA_Q_SERVICE_INTERVAL_EVENT | MQLONG | 队列服务时间间隔事件的控制属性 | 非 z/OS |
| MQIA_Q_TYPE | MQLONG | 队列类型 | |
| MQIA_QSG_DISP | MQLONG | 队列共享组处置 | z/OS |
| MQIA_RETENTION_INTERVAL | MQLONG | 队列保留时间间隔 | |
| MQIA_SCOPE | MQLONG | 队列定义作用域 | 非 z/OS |
| MQIA_SHAREABILITY | MQLONG | 是否可以共享队列以进行输入 | |
| MQIA_STATISTICS_Q | MQLONG | 控制队列的统计数据收集 | 非 z/OS |
| MQIA_TRIGGER_CONTROL | MQLONG | 触发器控制 | |
| MQIA_TRIGGER_DEPTH | MQLONG | 触发器深度 | |
| MQIA_TRIGGER_MSG_PRIORITY | MQLONG | 触发器的阈值消息优先级 | |
| MQIA_TRIGGER_TYPE | MQLONG | 触发器类型 | |
| MQIA_USAGE | MQLONG | 用法 | |

| 选择器 | 字段长度 | 描述 | 注 |
|----------------------|-------------------------|-----------|------|
| MQCA_ALTERATION_DATE | MQ_DATE_LENGTH | 最近一次更改日期 | |
| MQCA_ALTERATION_TIME | MQ_TIME_LENGTH | 最近一次更改的时间 | |
| MQCA_NAMELIST_DESC | MQ_NAMELIST_DESC_LENGTH | 名称列表描述 | |
| MQCA_NAMELIST_NAME | MQ_NAMELIST_NAME_LENGTH | 名称列表对象的名称 | |
| MQIA_NAMELIST_TYPE | MQLONG | 名称列表类型 | z/OS |

| 选择器 | 字段长度 | 描述 | 注 |
|-----------------|-------------------------------|-----------|------|
| MQCA_NAMES | MQ_Q_NAME_LENGTH × 列表中的名称数 | 名称列表中的名称 | |
| MQIA_NAME_COUNT | MQLONG | 名称列表中的名称数 | |
| MQIA_QSG_DISP | MQLONG | 队列共享组处置 | z/OS |

| 选择器 | 字段长度 | 描述 | 注 |
|----------------------|-----------------------------|-----------|------|
| MQCA_ALTERATION_DATE | MQ_DATE_LENGTH | 最近一次更改日期 | |
| MQCA_ALTERATION_TIME | MQ_TIME_LENGTH | 最近一次更改的时间 | |
| MQCA_APPL_ID | MQ_PROCESS_APPL_ID_LENGTH | 应用程序标识 | |
| MQCA_ENV_DATA | MQ_PROCESS_ENV_DATA_LENGTH | 环境数据 | |
| MQCA_PROCESS_DESC | MQ_PROCESS_DESC_LENGTH | 进程定义的描述 | |
| MQCA_PROCESS_NAME | MQ_PROCESS_NAME_LENGTH | 进程定义的名称 | |
| MQCA_USER_DATA | MQ_PROCESS_USER_DATA_LENGTH | 用户数据 | |
| MQIA_APPL_TYPE | MQLONG | 应用程序类型 | |
| MQIA_QSG_DISP | MQLONG | 队列共享组处置 | z/OS |

| 选择器 | 字段长度 | 描述 | 注 |
|----------------------------|---------------------|----------------|---|
| MQCA_ALTERATION_DATE | MQ_DATE_LENGTH | 最近一次更改日期 | |
| MQCA_ALTERATION_TIME | MQ_TIME_LENGTH | 最近一次更改的时间 | |
| MQCA_CHANNEL_AUTO_DEF_EXIT | MQ_EXIT_NAME_LENGTH | 自动通道定义出口名称 | |
| MQCA_CHINIT_SERVICE_PARM | | 保留供 IBM 使用 | |
| MQCA_CLUSTER_WORKLOAD_DATA | MQ_EXIT_DATA_LENGTH | 传递到集群工作负载出口的数据 | |
| MQCA_CLUSTER_WORKLOAD_EXIT | MQ_EXIT_NAME_LENGTH | 集群工作负载出口的名称 | |
| MQCA_COMMAND_INPUT_Q_NAME | MQ_Q_NAME_LENGTH | 系统命令输入队列名称 | |
| MQCA_DEAD_LETTER_Q_NAME | MQ_Q_NAME_LENGTH | 死信队列的名称 | |
| MQCA_DEF_XMIT_Q_NAME | MQ_Q_NAME_LENGTH | 缺省传输队列名称 | |

| 表 570: 队列管理器的 MQINQ 属性选择器 (继续) | | | |
|--------------------------------|-----------------------------|---|-----------------|
| 选择器 | 字段长度 | 描述 | 注 |
| MQCA_DNS_GROUP | MQ_DNS_GROUP_NAME_LENGTH | 用于处理要连接的队列共享组的入站传输的 TCP 侦听器的组名。使用工作负载管理器动态域名服务时, 该名称适用。 | z/OS |
| MQCA_IGQ_USER_ID | MQ_USER_ID_LENGTH | 组内排队用户标识 | z/OS |
| MQCA_INSTALLATION_DESC | MQ_INSTALLATION_DESC_LENGTH | 关联安装的描述 | 不是 z/OS。非 IBM i |
| MQCA_INSTALLATION_NAME | MQ_INSTALLATION_NAME_LENGTH | 与队列管理器关联的安装在名称 | 不是 z/OS。非 IBM i |
| MQCA_INSTALLATION_PATH | MQ_INSTALLATION_PATH_LENGTH | 安装关联 IBM WebSphere MQ 的路径 | 不是 z/OS。非 IBM i |
| MQCA_LU_GROUP_NAME | MQ_LU_NAME_LENGTH | 用于处理队列共享组要使用的入站传输的 LU 6.2 侦听器的通用 LU 名 | z/OS |
| MQCA_LU_NAME | MQ_LU_NAME_LENGTH | 要用于出站 LU 6.2 传输的 LU 的名称。将此名称设置为侦听器用于入站传输的同一 LU | z/OS |
| MQCA_LU62_ARM_SUFFIX | MQ_ARM_SUFFIX_LENGTH | SYS1.PARMLIB member APPCPMxxx 的后缀, 用于指定此通道启动程序的 LUADD | z/OS |
| MQCA_PARENT | MQ_Q_MGR_NAME_LENGTH | 指定为此队列管理器的父代的分层连接的队列管理器的名称 | |
| MQCA_Q_MGR_DESC | MQ_Q_MGR_DESC_LENGTH | 队列管理器描述 | |
| MQCA_Q_MGR_IDENTIFIER | MQ_Q_MGR_IDENTIFIER_LENGTH | 队列管理器标识 (H) | |
| MQCA_Q_MGR_NAME | MQ_Q_MGR_NAME_LENGTH | 本地队列管理器的名称 | |
| MQCA_QSG_NAME | MQ_QSG_NAME_LENGTH | 队列共享组名称 | z/OS |
| MQCA_REPOSITORY_NAME | MQ_CLUSTER_NAME_LENGTH | 队列管理器为其提供存储库服务的集群的名称 | |
| MQCA_REPOSITORY_NAMELIST | MQ_NAMELIST_NAME_LENGTH | 名称列表对象的名称, 其中包含队列管理器为其提供存储库服务的集群的名称 | |
| MQCA_TCP_NAME | MQ_TCP_NAME_LENGTH | 您正在使用的 TCP/IP 系统的名称 | z/OS |
| MQIA_ACCOUNTING_CONN_OVERRIDE | MQLONG | 覆盖记帐设置 | 非 z/OS |
| MQIA_ACCOUNTING_INTERVAL | MQLONG | 写入中间记帐记录的频率 | 非 z/OS |
| MQIA_ACCOUNTING_MQI | MQLONG | 控制 MQI 数据的记帐信息收集 | 非 z/OS |

| 表 570: 队列管理器的 MQINQ 属性选择器 (继续) | | | |
|--------------------------------|--------|--|--------|
| 选择器 | 字段长度 | 描述 | 注 |
| MQIA_ACCOUNTING_Q | MQLONG | 控制队列的记帐信息收集 | 非 z/OS |
| MQIA_ACTIVE_CHANNELS | MQLONG | 可随时处于活动状态的最大通道数 | z/OS |
| MQIA_ADOPTNEWMCA_CHECK | MQLONG | 为确定是否采用 MCA 而检查的元素。当检测到与已处于活动状态的 MCA 同名的新入站通道时，将执行此检查。 | z/OS |
| MQIA_ADOPTNEWMCA_INTERVAL | MQLONG | 新通道等待孤立通道结束的时间量 (以秒计) | 非 z/OS |
| MQIA_ADOPTNEWMCA_TYPE | MQLONG | 当检测到与 AdoptNewMCACheck 参数匹配的新入站通道请求时，是否自动重新启动特定通道类型的 MCA 的孤立实例 | z/OS |
| MQIA_AUTHORITY_EVENT | MQLONG | 权限事件的控制属性 | 非 z/OS |
| MQIA_BRIDGE_EVENT | MQLONG | IMS 网桥事件的控制属性 | z/OS |
| MQIA_CHANNEL_AUTO_DEF | MQLONG | 自动通道定义的控制属性 | 非 z/OS |
| MQIA_CHANNEL_AUTO_DEF_EVENT | MQLONG | 自动通道定义事件的控制属性 | 非 z/OS |
| MQIA_CHANNEL_EVENT | MQLONG | 通道事件的控制属性 | |
| MQIA_CHINIT_ADAPTERS | MQLONG | 用于处理 IBM WebSphere MQ 调用的适配器子任务数 | z/OS |
| MQIA_CHINIT_DISPATCHERS | MQLONG | 要用于通道启动程序的分派器数 | z/OS |
| MQIA_CHINIT_TRACE_AUTO_START | MQLONG | 是否自动启动通道启动程序跟踪 | z/OS |
| MQIA_CHINIT_TRACE_TABLE_SIZE | MQLONG | 通道启动程序的跟踪数据空间大小 (MB) | z/OS |
| MQIA_CLUSTER_WORKLOAD_LENGTH | MQLONG | 集群工作负载长度。 | |
| MQIA_CLWL_MRU_CHANNELS | MQLONG | 最近用于集群工作负载均衡的通道数 | |
| MQIA_CLWL_USEQ | MQLONG | 使用远程队列 | |
| MQIA_CODED_CHAR_SET_ID | MQLONG | 编码字符集标识 | |
| MQIA_COMMAND_EVENT | MQLONG | 命令事件的控制属性 | |
| MQIA_COMMAND_LEVEL | MQLONG | 队列管理器支持的命令级别 | |
| MQIA_CONFIGURATION_EVENT | MQLONG | 配置事件的控制属性 | 非 z/OS |
| MQIA_DEF_CLUSTER_XMIT_Q_TYPE | MQLONG | 要用于集群发送方通道的缺省传输队列类型。 | 非 z/OS |
| MQIA_DIST_LISTS | MQLONG | 分发列表支持 | 非 z/OS |

| 表 570: 队列管理器的 MQINQ 属性选择器 (继续) | | | |
|--------------------------------|--------|---|--------|
| 选择器 | 字段长度 | 描述 | 注 |
| MQIA_DNS_WLM | MQLONG | 处理队列共享组的入站传输的 TCP 侦听器是否向 Dynamic Domain Name Services 的工作负载管理器注册 | z/OS |
| MQIA_EXPIRY_INTERVAL | MQLONG | 扫描到期消息之间的时间间隔 | z/OS |
| MQIA_GROUP_UR | MQLONG | 此队列管理器是否启用 GROUP 恢复单元的控制属性。仅当队列管理器是队列共享组的成员时，GROUP 恢复单元处置才可用 | z/OS |
| MQIA_IGQ_PUT_AUTHORITY | MQLONG | 组内排队放置权限 | z/OS |
| MQIA_INHIBIT_EVENT | MQLONG | 禁止事件的控制属性 | 非 z/OS |
| MQIA_INTRA_GROUP_QUEUING | MQLONG | 组内排队支持 | z/OS |
| MQIA_LISTENER_TIMER | MQLONG | 当 APPC 或 TCP/IP 失败时，IBM WebSphere MQ 尝试重新启动侦听器之间的时间间隔 (以秒为单位)。 | z/OS |
| MQIA_LOCAL_EVENT | MQLONG | 本地事件的控制属性 | 非 z/OS |
| MQIA_LOGGER_EVENT | MQLONG | 禁止事件的控制属性 | 非 z/OS |
| MQIA_LU62_CHANNELS | MQLONG | 可以是当前通道或可以使用 LU 6.2 传输协议连接的客户端的最大通道数 | z/OS |
| MQIA_MSG_MARK_BROWSE_INTERVAL | MQLONG | 时间间隔 (以毫秒为单位)，在此时间间隔之后，队列管理器可以自动从浏览消息中除去标记。  注意: 不应将此值设置为低于缺省值 5000。 | |
| MQIA_MAX_CHANNELS | MQLONG | 可以是当前的最大通道数 (包括具有已连接客户端的服务器连接通道) | z/OS |
| MQIA_MAX_HANDLES | MQLONG | 最大句柄数 | |
| MQIA_MAX_MSG_LENGTH | MQLONG | 最大消息长度 | |
| MQIA_MAX_PRIORITY | MQLONG | 最高优先级 | |
| MQIA_MAX_UNCOMMITTED_MESSAGES | MQLONG | 工作单元中未落实的最大消息数 | |
| MQIA_OUTBOUND_PORT_MAX | MQLONG | 通过 MQIA_OUTBOUND_PORT_MIN, 定义绑定传出通道时要使用的端口号范围 | z/OS |
| MQIA_OUTBOUND_PORT_MIN | MQLONG | 通过 MQIA_OUTBOUND_PORT_MAX, 定义绑定传出通道时要使用的端口号范围 | z/OS |
| MQIA_PERFORMANCE_EVENT | MQLONG | 性能事件的控制属性 | 非 z/OS |
| MQIA_PLATFORM | MQLONG | 队列管理器所在的平台 | |

| 表 570: 队列管理器的 MQINQ 属性选择器 (继续) | | | |
|-----------------------------------|--------|---|--------|
| 选择器 | 字段长度 | 描述 | 注 |
| MQIA_PROT_POLICY_CAPABILITY | MQLONG | 指示 WebSphere MQ Advanced Message Security 的安全功能是否可用于队列管理器。 | |
| MQIA_PUBSUB_MAXMSG_RETRY_COUNT | MQLONG | 在同步点下尝试重新处理失败命令消息的次数 | |
| MQIA_PUBSUB_MODE | MQLONG | 发布/预订引擎和排队的发布/预订接口是否正在运行。要使用应用程序编程接口发布或预订的应用程序需要发布/预订引擎。由已排队的发布/预订接口监视的队列要求已排队的发布/预订接口正在运行。 | |
| MQIA_PUBSUB_NP_MSG | MQLONG | 是废弃 (还是保留) 未传递的输入消息 | |
| MQIA_PUBSUB_NP_RESP | MQLONG | 控制未传送响应消息的行为。 | |
| MQIA_PUBSUB_SYNC_PT | MQLONG | 是否仅在同步点下处理持久 (或所有) 消息 | |
| MQIA_QMGR_CFCONLOS | MQLONG | 指定当队列管理器与管理结构或将 CFCONLOS 设置为 ASQMGR 的任何 CF 结构失去连接时要执行的操作 | z/OS |
| MQIA_RECEIVE_TIMEOUT | MQLONG | TCP/IP 通道等待从其伙伴接收数据 (包括脉动信号) 的大约时间长度, 然后再返回到不活动状态。该值为数字, 由 MQIA_RECEIVE_TIMEOUT_TYPE 限定。 | z/OS |
| MQIA_RECEIVE_TIMEOUT_MIN | MQLONG | TCP/IP 通道在返回到不活动状态之前等待从其合作伙伴接收数据 (包括脉动信号) 的最短时间 | z/OS |
| MQIA_RECEIVE_TIMEOUT_TYPE | MQLONG | TCP/IP 通道等待从其伙伴接收数据 (包括脉动信号) 的大约时间长度, 然后再返回到不活动状态。 MQIA_RECEIVE_TIMEOUT_TYPE 是应用于 MQIA_RECEIVE_TIMEOUT 的限定符。 | z/OS |
| MQIA_REMOTE_EVENT | MQLONG | 远程事件的控制属性 | 非 z/OS |
| MQIA_SECURITY_CASE | MQLONG | 安全概要文件的情况 | z/OS |
| MQIA_SSL_EVENT | MQLONG | 通道事件的控制属性 | |
| MQIA_SSL_FIPS_REQUIRED | MQLONG | 仅将 FIPS 认证的算法用于密码术 | |
| MQIA_SSL_RESET_COUNT | MQLONG | SSL 密钥重新设置计数 | |
| MQIA_START_STOP_EVENT | MQLONG | 启动停止事件的控制属性 | 非 z/OS |
| MQIA_STATISTICS_AUTO_CLUSTER_SSDR | MQLONG | 控制集群发送方通道的统计信息监视信息收集 | 非 z/OS |
| MQIA_STATISTICS_CHANNEL | MQLONG | 控制通道的统计数据收集 | 非 z/OS |

| 表 570: 队列管理器的 MQINQ 属性选择器 (继续) | | | |
|--------------------------------|--------|---|--------|
| 选择器 | 字段长度 | 描述 | 注 |
| MQIA_STATISTICS_INTERVAL | MQLONG | 写入统计信息监视数据的频率 | 非 z/OS |
| MQIA_STATISTICS_MQI | MQLONG | 控制队列管理器的统计信息监视信息的收集 | 非 z/OS |
| MQIA_STATISTICS_Q | MQLONG | 控制队列的统计数据收集 | 非 z/OS |
| MQIA_SYNCPOINT | MQLONG | 同步点可用性 | |
| MQIA_TCP_CHANNELS | MQLONG | 使用 TCP/IP 传输协议可以是当前通道或可以连接的客户端的最大通道数 | z/OS |
| MQIA_TCP_KEEP_ALIVE | MQLONG | 是否使用 TCP KEEPALIVE 工具来检查连接的另一端是否仍然可用 | z/OS |
| MQIA_TCP_STACK_TYPE | MQLONG | 通道启动程序是只能使用 TCPNAME 中指定的 TCP/IP 地址空间, 还是可以选择绑定到任何所选 TCP/IP 地址 | z/OS |
| MQIA_TRACE_ROUTE_RECORDING | MQLONG | 控制跟踪路由信息的记录 | z/OS |
| MQIA_TREE_LIFE_TIME | MQLONG | 未使用的非管理主题的生存期 | |
| MQIA_TRIGGER_INTERVAL | MQLONG | 触发器时间间隔 | |

IntAttrCount

类型: MQLONG -输入

这是 *IntAttrs* 数组中的元素数。零是有效值。

如果 *IntAttrCount* 至少是 *Selectors* 参数中 MQIA_* 选择器的数目, 那么将返回请求的所有整数属性。

IntAttrs

类型: MQLONG × *IntAttrCount* -输出

这是 *IntAttrCount* 整数属性值的数组。

返回整数属性值的顺序与 *Selectors* 参数中的 MQIA_* 选择器相同。如果数组包含的元素数超过 MQIA_* 选择器的数目, 那么多余的元素保持不变。

如果 *Hobj* 表示队列, 但属性选择器不适用于该类型的队列, 那么将返回特定值 MQIAV_NOT_APPLICABLE。将针对 *IntAttrs* 数组中的相应元素返回此值。

如果 *IntAttrCount* 或 *SelectorCount* 参数为零, 那么不会引用 *IntAttrs*。在这种情况下, 以 C 或 S/390 汇编程序编写的程序传递的参数地址可能为空。

CharAttr 长度

类型: MQLONG -输入

这是 *CharAttrs* 参数的长度 (以字节计)。

CharAttrLength 必须至少是所请求字符属性的长度总和 (请参阅 *Selectors*)。零是有效值。

CharAttrs

类型: MQCHAR × *CharAttrLength* -输出

这是在其中返回字符属性并将其并置在一起的缓冲区。缓冲区的长度由 *CharAttrLength* 参数提供。

返回字符属性的顺序与 *Selectors* 参数中的 MQCA_* 选择器相同。每个属性字符串的长度对于每个属性都是固定的 (请参阅 *Selectors*), 如果需要, 会将其中的值用空格填充到右边。您可以提供一个大

于所需大小的缓冲区，以包含所有请求的字符属性和填充。返回的超出最后一个属性值的字节保持不变。

如果 *Hobj* 表示队列，但属性选择器不适用于该类型的队列，那么将返回完全由星号 (*) 组成的字符串。将返回星号作为 *CharAttr* 中该属性的值。

如果 *CharAttrLength* 或 *SelectorCount* 参数为零，那么不会引用 *CharAttr*。在这种情况下，以 C 或 S/390 汇编程序编写的程序传递的参数地址可能为空。

CompCode

类型: MQLONG - 输出

完成代码:

MQCC_OK

成功完成。

MQCC_WARNING

警告（部分完成）。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 无理由报告。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

(2008, X'7D8') 不允许为字符属性提供足够的空间。

MQRC_INT_ATTR_COUNT_TOO_SMALL

(2022, X'7E6') 整数属性不允许有足够的空间。

MQRC_SELECTOR_NOT_FOR_TYPE

(2068, X'814') 选择器不适用于队列类型。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID 和主 ASID 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_CF_STRUC_FAILED

(2373, X'945') 耦合设施结构失败。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') 正在使用耦合设施结构。

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6') 字符属性的长度无效。

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7') 字符属性字符串无效。

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CONNECTION_BROKEN
(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') 未授权进行连接。

MQRC_CONNECTION_STOPPING
(2203, X'89B') 连接正在关闭。

MQRC_HCONN_ERROR
(2018, X'7E2') 连接句柄无效。

MQRC_HOBJ_ERROR
(2019, X'7E3') 对象句柄无效。

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') 整数属性计数无效。

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') 整数属性数组无效。

MQRC_NOT_OPEN_FOR_INQUIRE
(2038, X'7F6') 队列未打开以进行查询。

MQRC_OBJECT_CHANGED
(2041, X'7F9') 对象定义自打开以来已更改。

MQRC_OBJECT_DAMAGED
(2101, X'835') 对象已损坏。

MQRC_PAGESET_ERROR
(2193, X'891') 访问页集数据集时出错。

MQRC_Q_DELETED
(2052, X'804') 已删除队列。

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') 队列管理器名称无效或未知。

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') 队列管理器不可用于连接。

MQRC_Q_MGR_STOPPING
(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM
(2102, X'836') 系统资源不足。

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') 选择器计数无效。

MQRC_SELECTOR_ERROR
(2067, X'813') 属性选择器无效。

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') 选择器计数过大。

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') 存储空间不足。

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') 出口程序禁止调用。

MQRC_UNEXPECTED_ERROR
(2195, X'893') 发生意外错误。

有关这些代码的详细信息; 请参阅 [原因码](#)

使用说明

1. 返回的值是所选属性的快照。在应用程序可以对返回的值执行操作之前，不保证属性保持不变。
2. 打开模型队列时，将创建动态本地队列。即使打开模型队列以查询其属性，也会创建动态本地队列。
动态队列的属性在很大程度上与创建动态队列时模型队列的属性相同。如果随后在此队列上使用 MQINQ 调用，那么队列管理器将返回动态队列的属性，而不是模型队列的属性。有关动态队列继承模型队列的哪些属性的详细信息，请参阅第 733 页的表 573。
3. 如果要查询的对象是别名队列，那么 MQINQ 调用返回的属性值是别名队列的属性。这些属性不是别名解析为的基本队列或主题的属性。
4. 如果要查询的对象是集群队列，那么可查询的属性取决于队列的打开方式：

- 您可以打开集群队列以进行查询以及一个或多个输入，浏览或设置操作。要执行此操作，必须存在集群队列的本地实例才能成功打开。在这种情况下，可以查询的属性是对本地队列有效的属性。

如果在未指定输入，浏览或设置的情况下打开集群队列进行查询，那么如果尝试查询仅对本地队列有效的属性，而不对集群队列有效的属性，那么调用将返回完成代码 MQCC_WARNING 和原因码 MQRC_SELECTOR_NOT_FOR_TYPE (2068)。

- 您可以在传递所连接队列管理器的基本队列管理器名称时打开集群队列以进行查询。

要执行此操作，必须存在集群队列的本地实例才能成功打开。如果未传递基本队列管理器，那么如果尝试查询仅对本地队列有效而对集群队列无效的属性，那么调用将返回完成代码 MQCC_WARNING 和原因码 MQRC_SELECTOR_NOT_FOR_TYPE (2068)。

- 如果单独打开集群队列以进行查询或查询和输出，那么只能查询列出的属性。在此情况下，QType 属性的值为 MQQT_CLUSTER：

- MQCA_Q_DESC
- MQCA_Q_NAME
- MQIA_DEF_BIND
- MQIA_DEF_PERSISTENCE
- MQIA_DEF_PRIORITY
- MQIA_INHIBIT_PUT
- MQIA_Q_TYPE

您可以在没有固定绑定的情况下打开集群队列。您可以使用 MQOPEN 调用上指定的 MQOO_BIND_NOT_FIXED 将其打开。或者，指定 MQOO_BIND_AS_Q_DEF，并将队列的 DefBind 属性设置为 MQBND_BIND_NOT_FIXED。如果打开没有固定绑定的集群队列，那么该队列的连续 MQINQ 调用可能会查询该集群队列的不同实例。但是，对于所有具有相同属性值的实例，这是典型情况。

- 可以为集群定义别名队列对象。由于 TARGTYPE 和 TARGET 不是集群属性，因此在别名队列上执行 MQOPEN 进程的进程无法识别别名解析到的对象。

在初始 MQOPEN 期间，别名队列解析为集群中的队列管理器和队列。在远程队列管理器上再次进行名称解析，在此解析别名队列的 TARGTYPE。

如果别名队列解析为主题别名，那么将在此远程队列管理器上发布放入别名队列的消息。

请参阅 [集群队列](#)

5. 您可能想要查询多个属性，然后使用 MQSET 调用来设置其中一些属性。要高效地进行程序查询和设置，请将要设置的属性放在选择器数组的开头。如果执行此操作，那么可以将计数减少的相同数组用于 MQSET。
6. 如果出现多个警告情境 (请参阅 *CompCode* 参数)，那么返回的原因码是以下列表中适用的第一个原因码：
 - a. MQRC_SELECTOR_NOT_FOR_TYPE
 - b. MQRC_INT_ATTR_COUNT_TOO_SMALL
 - c. MQRC_CHAR_ATTRS_TOO_SHORT
7. 以下主题包含有关对象属性的信息：
 - [第 731 页的『队列的属性』](#)

- [第 759 页的『名称列表的属性』](#)
- [第 761 页的『进程定义的属性』](#)
- [第 701 页的『队列管理器的属性』](#)

C 调用

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
       CharAttrLength, CharAttrs, &CompCode, &Reason);
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                  INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                  CHARATTRS, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.
```

PL/I 调用

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

按如下所示声明参数:

```
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 */

```

高级汇编程序调用

```

CALL MQINQ, (HCONN, HOBJ, SELECTORCOUNT, SELECTORS, INTATTRCOUNT, X
            INTATTRS, CHARATTRLENGTH, CHARATTRS, COMPCODE, REASON)

```

按如下所示声明参数:

| | | | |
|----------------|----|-------|---------------------------------------|
| 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 |

Visual Basic 调用

```

MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, CompCode, Reason

```

按如下所示声明参数:

```

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-查询消息属性

MQINQMP 调用返回消息属性的值。

语法

```

MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type, ValueLength, Value,
DataLength, CompCode, Reason)

```

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。Hconn 的值必须与用于创建 Hmsg 参数中指定的消息句柄的连接句柄相匹配。

如果消息句柄是使用 MQHC_UNASSOCIATED_HCONN 创建的，那么必须在查询消息句柄的属性的线程上建立有效连接，否则调用将失败并返回 MQRC_CONNECTION_BROKEN。

Hmsg

类型:MQHMSG-输入

这是要查询的消息句柄。该值由先前的 MQCRTMH 调用返回。

InqProp 选项

类型:MQIMPO-输入/输出

请参阅 [MQIMPO](#) 数据类型以获取详细信息。

名称

类型:MQCHARV-输入/输出

要查询的属性的名称。

如果找不到具有此名称的属性，那么调用将失败，原因为 MQRC_PROPERTY_NOT_AVAILABLE。

可以在属性名末尾使用通配符百分号 (%)。通配符与零个或多个字符匹配，包括句点 (.) 字符。这允许应用程序查询许多属性的值。使用选项 MQIMPO_INQ_FIRST 调用 MQINQMP 以获取第一个匹配属性，使用选项 MQIMPO_INQ_NEXT 再次调用 MQINQMP 以获取下一个匹配属性。如果没有更多匹配属性可用，那么调用将失败并返回 MQRC_PROPERTY_NOT_AVAILABLE。如果 InqPropOpts 结构的 ReturnedName 字段是使用返回的属性名称的地址或偏移量初始化的，那么将在从 MQINQMP 返回时使用已匹配的属性名称完成此操作。如果 InqPropOpts 结构中 ReturnedName 的 VSBufSize 字段小于返回的属性名的长度，那么将使用原因 MQRC_PROPERTY_NAME_TOO_BIG 设置完成代码 MQCC_FAILED。

将返回具有已知同义词的属性，如下所示：

1. 带有前缀 "mqps" 的属性。作为 WebSphere MQ 属性名返回。例如，"MQTopicString" 是返回的名称，而不是 "mqps.Top"
2. 带有前缀 "jms" 的属性。或 "mcd"。作为 JMS 头字段名称返回，例如，"JMSExpiration" 是返回的名称，而不是 "jms.Exp"。
3. 带有前缀 "usr." 的属性 返回时不带该前缀，例如，返回 "Color" 而不是 "usr.Color"。

带有同义词的属性仅返回一次。

在 C 编程语言中，定义了以下宏变量以查询所有属性，然后查询以 "usr." 开头的的所有属性：

MQPROP_INQUIRE_ALL

查询消息的所有属性。

MQPROP_INQUIRE_ALL 可通过以下方式使用：

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR

查询启动 "usr." 的消息的所有属性。返回的名称不带 "usr"。前缀。

如果指定了 MQIMP_INQ_NEXT，但自上次调用以来名称已更改，或者这是第一次调用，那么将隐含 MQIMPO_INQ_FIRST。

请参阅 [属性名](#) 和 [属性名限制](#)，以获取有关使用属性名的更多信息。

PropDesc

类型:MQPD-输出

此结构用于定义属性的属性，包括不支持该属性时发生的情况，该属性所属的消息上下文以及应该将该属性复制到的消息。请参阅 [MQPD](#) 以获取此结构的详细信息。

类型

类型:MQLONG-输入/输出

从 MQINQMP 调用返回时，此参数设置为数据类型值。数据类型可以是下列任何一项：

MQTYPE_BOOLEAN

布尔值。

MQTYPE_BYTE_STRING

字节字符串。

MQTYPE_INT8

8 位带符号整数。

MQTYPE_INT16

16 位带符号整数。

MQTYPE_INT32

32 位带符号整数。

MQTYPE_INT64

64 位带符号整数。

MQTYPE_FLOAT32

32 位浮点数。

MQTYPE_FLOAT64

64 位浮点数。

MQTYPE_STRING

字符串。

MQTYPE_NULL

该属性存在，但具有空值。

如果无法识别属性值的数据类型，那么将返回 `MQTYPE_STRING`，并将该值的字符串表示法放入值区域中。可以在 `InqPropOpts` 参数的 `TypeString` 字段中找到数据类型的字符串表示。将返回警告完成代码，原因为 `MQRC_PROP_TYPE_NOT_SUPPORTED`。

此外，如果指定了选项 `MQIMPO_CONVERT_TYPE`，那么将请求转换属性值。使用类型作为输入，以指定要作为属性返回的数据类型。有关数据类型转换的详细信息，请参阅 `MQIMPO` 结构的 `MQIMPO_CONVERT_TYPE` 选项的描述。

如果不请求类型转换，那么可以在输入上使用以下值：

MQTYPE_AS_SET

将返回该属性的值，而不转换其数据类型。

ValueLength

类型：`MQLONG`-输入

"值"区域的长度(以字节为单位)。为不需要返回值的属性指定零。这些属性可以是应用程序设计为具有空值或空字符串的属性。如果指定了 `MQIMPO_QUERY_LENGTH` 选项，那么还要指定零；在这种情况下，不会返回任何值。

Value

类型：`MQBYTE`*ValueLength* -输出

这是包含查询的属性值的区域。缓冲区应该在适合于所返回值的边界上对齐。未能执行此操作可能会导致稍后访问该值时发生错误。

如果 `ValueLength` 小于属性值的长度，那么会将尽可能多的属性值移动到值中，并且调用将失败，并返回完成代码 `MQCC_FAILED` 和原因 `MQRC_PROPERTY_VALUE_TOO_BIG`。

值中数据的字符集由 `InqPropOpts` 参数中的 `ReturnedCCSID` 字段提供。值中数据的编码由 `InqPropOpts` 参数中的 `ReturnedEncoding` 字段提供。

在 C 编程语言中，该参数被声明为指向 `void` 的指针；任何类型的数据的地址都可以被指定为该参数。

如果 `ValueLength` 参数为零，那么不会引用值，并且以 C 或 `System/390` 汇编程序编写的程序传递的值可以为空。

DataLength

类型：`MQLONG` - 输出

这是值区域中返回的实际属性值的长度(以字节计)。

如果 *DataLength* 小于属性值长度，那么在从 MQINQMP 调用返回时仍会填充 *DataLength*。这允许应用程序确定容纳属性值所需的缓冲区大小，然后使用相应大小的缓冲区重新发出调用。

还可以返回以下值。

如果 *Type* 参数设置为 MQTYPE_STRING 或 MQTYPE_BYTE_STRING:

MQVL_EMPTY_STRING

该属性存在，但不包含任何字符或字节。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_PROP_NAME_NOT_汇率

(2492, X'09BC') 未转换返回的属性名。

MQRC_PROP_VALUE_NOT_CONVERTED

(2466, X'09A2') 未转换属性值。

MQRC_PROP_TYPE_NOT_SUPPORTED

(2467, X'09A3') 不支持属性数据类型。

MQRC_RFH_FORMAT_ERROR

(2421, X'0975 ') 无法解析包含属性的 MQRFH2 文件夹。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'0852 ') 无法装入适配器服务模块。

MQRC_ASID_MISMATCH

(2157, X'086D') 主 ASID 和主 ASID 不同。

MQRC_BUFFER_ERROR

(2004, X'07D4') 值参数无效。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') 值长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009 年, X'07D9') 与队列管理器的连接丢失。

MQRC_DATA_LENGTH_ERROR

(2010, X'07DA') 数据长度参数无效。

MQRC_IMPO_ERROR

(2464, X'09A0') 查询消息属性选项结构无效。

MQRC_HMSG_ERROR

(2460, X'099C') 消息句柄无效。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') 消息句柄已在使用中。

MQRC_OPTIONS_ERROR

(2046, X'07F8') 选项无效或不一致。

MQRC_PD_ERROR

(2482, X'09B2') 属性描述符结构无效。

MQRC_PROP_CONV_NOT_SUPPORTED

(2470, X'09A6') 不支持从实际数据类型转换为请求的数据类型。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') 属性名无效。

MQRC_PROPERTY_NAME_TOO_BIG

(2465, X'09A1') 属性名对于返回的名称缓冲区太大。

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') 属性不可用。

MQRC_PROPERTY_VALUE_TOO_BIG

(2469, X'09A5') 属性值对于"值"区域过大。

MQRC_PROP_NUMBER_FORMAT_ERROR

(2472, X'09A8') 在值数据中迁到数字格式错误。

MQRC_PROPERTY_TYPE_ERROR

(2473, X'09A9') 请求的属性类型无效。

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') 属性名编码字符集标识无效。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'0871') 可用存储空间不足。

MQRC_UNEXPECTED_ERROR

(2195, X'0893') 发生意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

C 调用

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

按如下所示声明参数：

```
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 */
```

COBOL 调用

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDSC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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 CMQCHRVA.
** Property descriptor
01 PROPDSC.
   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.
```

PL/I 调用

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

按如下所示声明参数:

```
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 */
```

高级汇编程序调用

```
CALL MQINQMP, (HCONN, HMSG, INQMSGOPTS, NAME, PROPDSC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON)
```

按如下所示声明参数:

| | | | |
|-------------|----------|-------|--|
| HCONN | DS | F | Connection handle |
| HMSG | DS | D | Message handle |
| INQMSGOPTS | CMQIMPOA | , | Options that control the action of MQINQMP |
| NAME | CMQCHRVA | , | Property name |
| PROPDSC | CMQPDV | , | 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-将消息句柄转换为缓冲区

MQMHBUF 调用将消息句柄转换为缓冲区, 并且是 MQBUFMH 调用的逆调用。

语法

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Name*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。*Hconn* 的值必须与用于创建 *Hmsg* 参数中指定的消息句柄的连接句柄相匹配。

如果消息句柄是使用 MQHC_UNASSOCIATED_HCONN 创建的,那么必须在删除消息句柄的线程上建立有效连接。如果未建立有效连接,那么调用将失败,并返回 MQRC_CONNECTION_BROKEN。

Hmsg

类型:MQHMSG-输入

这是需要缓冲区的消息句柄。该值由先前的 MQCRTMH 调用返回。

MsgHBuf 选项

类型:MQMHBO-输入

MQMHBO 结构允许应用程序指定用于控制如何从消息句柄生成缓冲区的选项。

有关详细信息,请参阅第 412 页的『MQMHBO-消息句柄到缓冲区选项』。

Name

类型:MQCHARV-输入

要放入缓冲区中的一个或多个属性的名称。

如果找不到与名称匹配的属性,那么调用将失败并返回 MQRC_PROPERTY_NOT_AVAILABLE。

可以使用通配符将多个属性放入缓冲区中。为此,请在属性名末尾使用通配符 "%"。此通配符与零个或多个字符匹配,包括 "." 字符结尾。

在 C 编程语言中,定义了以下宏变量以查询所有属性以及以 "usr" 开头的所有属性:

MQPROP_INQUIRE_ALL

将消息的所有属性放入缓冲区

MQPROP_INQUIRE_ALL_USR

放入以字符 "usr." 开头的消息的所有属性。进入缓冲区。

请参阅 [属性名](#) 和 [属性名限制](#), 以获取有关使用属性名的更多信息。

MsgDesc

类型:MQMD-输入/输出

MsgDesc 结构描述缓冲区的内容。

在输出时, *Encoding*, *CodedCharSetId* 和 *Format* 字段设置为正确描述缓冲区中由调用写入的数据的编码,字符集标识和格式。

此结构中的数据采用应用程序的字符集和编码。

BufferLength

类型:MLONG-输入

BufferLength 是缓冲区的长度(以字节为单位)。

buffer

类型:MQBYTEExBuffer 长度-输出

Buffer 定义要包含消息属性的区域。必须在 4 字节边界上对齐缓冲区。

如果 *BufferLength* 小于在 *Buffer* 中存储属性所需的长度,那么 MQMHBUF 将失败,并返回 MQRC_PROPERTY_VALUE_TOO_BIG。

即使调用失败，缓冲区的内容也会更改。

DataLength

类型：MQLONG - 输出

DataLength 是缓冲区中返回的属性的长度 (以字节计)。如果值为零，那么没有任何属性与 *Name* 中给定的值匹配，并且调用失败，原因码为 MQRC_PROPERTY_NOT_AVAILABLE。

如果 *BufferLength* 小于在缓冲区中存储属性所需的长度，那么 MQMHBUF 调用将失败并返回 MQRC_PROPERTY_VALUE_TOO_BIG，但仍会在 *DataLength* 中输入值。这允许应用程序确定容纳属性所需的缓冲区大小，然后使用所需的 *BufferLength* 重新发出调用。

CompCode

类型：MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED：

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_MHBO_ERROR

(2501, X'095C') 缓冲区选项结构的消息句柄无效。

MQRC_BUFFER_ERROR

(2004, X'07D4') 缓冲区参数无效。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') 与队列管理器的连接丢失。

MQRC_DATA_LENGTH_ERROR

(2010, X'07DA') 数据长度参数无效。

MQRC_HMSG_ERROR

(2460, X'099C') 消息句柄无效。

MQRC_MD_ERROR

(2026, X'07EA') 消息描述符无效。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') 消息句柄已在使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') 选项无效或不一致。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') 属性名无效。

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') 属性不可用。

MQRC_PROPERTY_VALUE_TOO_BIG

(2469, X'09A5') BufferLength 值太小, 无法包含指定的属性。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息, 请参阅[原因码](#)。

C 调用

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,  
&DataLength, &CompCode, &Reason);
```

按如下所示声明参数:

```
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 */
```

使用说明

MQMHBUF 将消息句柄转换为缓冲区。

您可以将其与 MQGET API 出口配合使用, 以使用消息属性 API 来访问某些属性, 然后将这些属性在缓冲区中传递回设计为使用 MQRFH2 头而不是消息句柄的应用程序。

此调用是 MQBUFMH 调用的逆调用, 可用于将消息属性从缓冲区解析为消息句柄。

COBOL 调用

```
CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,  
BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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 CMQMHBV.  
** 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.

```

PL/I 调用

```

call MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
              DataLength, CompCode, Reason);

```

按如下所示声明参数:

```

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 */

```

高级汇编程序调用

```

CALL MQMHBUF, (HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
              BUFFER,DATALENGTH,COMPCODE,REASON)

```

按如下所示声明参数:

| | | | |
|--------------|----------|-------|--|
| 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-打开对象

MQOPEN 调用建立对对象的访问权。

以下类型的对象有效:

- 队列 (包括分发列表)
- 名称列表
- 进程定义
- 队列管理器
- Topic

语法

```

MQOPEN (Hconn, ObjDesc, 选项, Hobj, CompCode, Reason)

```

参数

Hconn

类型 :MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 Hconn 的值。

在 z/OS (对于 CICS 应用程序) 和 IBM i (对于以兼容性方式运行的应用程序) 上, 可以省略 MQCONN 调用, 并为 Hconn 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

ObjDesc

类型 :MQOD-输入/输出

这是用于标识要打开的对象的结构; 请参阅第 414 页的『MQOD-对象描述符』以获取详细信息。

如果 ObjDesc 参数中的 ObjectName 字段是模型队列的名称, 那么是动态本地队列是使用模型队列的属性创建的; 无论您在 Options 参数上指定的选项如何, 都会发生此情况。使用 MQOPEN 调用返回的 Hobj 的后续操作将在新的动态队列上执行, 而不是在模型队列上执行。即使对于 MQINQ 和 MQSET 调用也是如此。ObjDesc 参数中模型队列的名称将替换为创建的动态队列的名称。动态队列的类型由模型队列的 DefinitionType 属性的值确定 (请参阅第 731 页的『队列的属性』)。有关适用于动态队列的关闭选项的信息, 请参阅 MQCLOSE 调用的描述。

options

类型 :MQLONG-输入

必须至少指定下列其中一个选项:

- MQOO_BROWSE
- MQOO_INPUT_* (仅其中一个)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_* (仅其中一个)

请参阅下表以获取这些选项的详细信息; 可以根据需要指定其他选项。如果需要多个选项, 那么值可以是:

- 一起添加 (请勿多次添加同一常量), 或者
- 通过按位“或”运算组合在一起 (如果编程语言支持位运算)。

将记录无效的组合; 所有其他组合都有效。只允许使用适用于 ObjDesc 指定的对象类型的选项。下表显示了查询和主题的有效 MQOPEN 选项。

| 选项 | 别名 ¹ | 本地和模型 | 远程 | 非本地集群 | 通讯组列表 | Topic |
|---|-----------------|-------|----------|-------|-------|-------|
| <u>MQOO_INPUT_AS_Q_DEF</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_INPUT_SHARED</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_INPUT_EXCLUSIVE</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_OUTPUT</u> | Yes | Yes | Yes | Yes | Yes | Yes |
| <u>MQOO_BROWSE</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_CO_OP</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_INQUIRE</u> | Yes | Yes | <u>2</u> | Yes | 否 | 否 |
| <u>MQOO_SET</u> | Yes | Yes | <u>2</u> | 否 | 否 | 否 |
| <u>MQOO_BIND_ON_OPEN</u> ³ | Yes | Yes | Yes | Yes | Yes | 否 |
| <u>MQOO_BIND_NOT_FIXED</u> ³ | Yes | Yes | Yes | Yes | Yes | 否 |
| <u>MQOO_BIND_ON_GROUP</u> ³ | Yes | Yes | Yes | Yes | Yes | 否 |

| 选项 | 别名 ¹ | 本地和模型 | 远程 | 非本地集群 | 通讯组列表 | Topic |
|--|-----------------|-------|-----|-------|-------|----------|
| <u>MQOO_BIND_AS_Q_DEF</u> ³ | Yes | Yes | Yes | Yes | Yes | 否 |
| <u>MQOO_SAVE_ALL_CONTEXT</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_PASS_IDENTITY_CONTEXT</u> | Yes | Yes | Yes | Yes | Yes | <u>4</u> |
| <u>MQOO_PASS_ALL_CONTEXT</u> | Yes | Yes | Yes | Yes | Yes | Yes |
| <u>MQOO_SET_IDENTITY_CONTEXT</u> | Yes | Yes | Yes | Yes | Yes | <u>4</u> |
| <u>MQOO_SET_ALL_CONTEXT</u> | Yes | Yes | Yes | Yes | Yes | Yes |
| <u>MQOO_NO_READ_AHEAD</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_READ_AHEAD</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_READ_AHEAD_AS_Q_DEF</u> | Yes | Yes | 否 | 否 | 否 | 否 |
| <u>MQOO_ALTERNATE_USER_AUTHORITY</u> | Yes | Yes | Yes | Yes | Yes | Yes |
| <u>MQOO_FAIL_IF_QUIESCING</u> | Yes | Yes | Yes | Yes | Yes | Yes |
| <u>MQOO_RESOLVE_LOCAL_Q</u> | Yes | Yes | Yes | Yes | 否 | 否 |
| <u>MQOO_RESOLVE_LOCAL_TOPIC</u> | 否 | 否 | 否 | 否 | 否 | Yes |
| <u>MQOO_NO_多点广播</u> | 否 | 否 | 否 | 否 | 否 | Yes |

注:

1. 别名选项的有效性取决于别名解析到的队列的选项的有效性。
2. 此选项仅对远程队列的本地定义有效。
3. 可以为任何队列类型指定此选项，但如果队列不是集群队列，那么将忽略此选项。但是，即使别名队列不在集群中，*DefBind* 队列属性也会覆盖基本队列。
4. 这些属性可与主题配合使用，但仅影响保留消息的上下文集，而不影响发送给任何订户的上下文字段。

访问选项: 以下选项控制可以对对象执行的操作的类型:

MQOO_INPUT_AS_Q_DEF

打开队列以使用队列定义的缺省值获取消息。

打开队列以与后续 MQGET 调用配合使用。访问类型为共享或互斥，具体取决于 *DefInputOpenOption* 队列属性的值; 请参阅 第 731 页的『队列的属性』以获取详细信息。

此选项仅对本地队列，别名队列和模型队列有效; 对于非队列的远程队列，分发列表和对象无效。

MQOO_INPUT_SHARED

打开队列以获取具有共享访问权的消息。

打开队列以与后续 MQGET 调用配合使用。如果队列当前由此应用程序或另一个使用 MQOO_INPUT_SHARED 的应用程序打开，但如果队列当前使用 MQOO_INPUT_EXCLUSIVE 打开，那么调用可能成功失败，原因码为 MQRC_OBJECT_IN_USE。

此选项仅对本地队列，别名队列和模型队列有效; 对于非队列的远程队列，分发列表和对象无效。

MQOO_INPUT_EXCLUSIVE

打开队列以获取具有独占访问权的消息。

打开队列以与后续 MQGET 调用配合使用。如果此应用程序或其他应用程序当前打开队列以用于任何类型的输入 (MQOO_INPUT_SHARED 或 MQOO_INPUT_EXCLUSIVE)，那么调用将失败，原因码为 MQRC_OBJECT_IN_USE。

此选项仅对本地队列，别名队列和模型队列有效; 对于非队列的远程队列，分发列表和对象无效。

MQOO_OUTPUT

打开队列以放置消息，或打开主题或主题字符串以发布消息。

打开队列或主题以用于后续 MQPUT 调用。

使用此选项的 MQOPEN 调用可能成功, 即使 *InhibitPut* 队列属性设置为 MQQA_PUT_ALLOWED (尽管在此属性设置为此值时后续 MQPUT 调用失败) 也是如此。

此选项对所有类型的队列 (包括分发列表和主题) 都有效。

以下说明适用于这些选项:

- 只能指定其中一个选项。
- 具有这些选项之一的 MQOPEN 调用可能成功, 即使 *InhibitGet* 队列属性设置为 MQQA_GET_ALLOWED (尽管后续 MQGET 调用在该属性设置为该值时失败)。
- 如果将队列定义为不可共享 (即, *Shareability* 队列属性的值为 MQQA_NOT_SHAREABLE), 那么打开队列以进行共享访问的尝试将视为尝试打开具有互斥访问权的队列。
- 如果使用其中一个选项打开别名队列, 那么针对别名解析到的基本队列进行独占使用 (或其他应用程序是否具有独占使用) 测试。
- 如果 *ObjectQMgrName* 是队列管理器别名的名称, 那么这些选项无效; 即使用于队列管理器别名判别的远程队列的本地定义中的 *RemoteQMgrName* 属性值是本地队列管理器的名称, 也是如此。

MQOO_BROWSE

打开队列以浏览消息。

将打开此队列以用于具有以下某个选项的后续 MQGET 调用:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

即使当前针对 MQOO_INPUT_EXCLUSIVE 打开了队列, 也允许执行此操作。带有 MQOO_BROWSE 选项的 MQOPEN 调用将建立浏览光标, 并将其逻辑地定位在队列上的第一条消息之前; 请参阅 [MQGMO-选项字段](#) 以获取更多信息。

此选项仅对本地队列, 别名队列和模型队列有效; 对于非队列的远程队列, 分发列表和对象无效。如果 *ObjectQMgrName* 是队列管理器别名的名称, 那么此属性也无效; 即使用于队列管理器别名判别的远程队列的本地定义中的 *RemoteQMgrName* 属性值是本地队列管理器的名称, 也是如此。

MQOO_CO_OP

作为一组手柄的协同成员打开。

此选项仅对 MQOO_BROWSE 选项有效。如果未指定 MQOO_BROWSE, 那么 MQOPEN 将返回 MQRC_OPTIONS_ERROR。

返回的句柄被视为具有下列其中一个选项的后续 MQGET 调用的协同句柄集的成员:

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

此选项仅对本地队列, 别名队列和模型队列有效; 对于非队列的远程队列, 分发列表和对象无效。

MQOO_INQUIRE

打开对象以查询属性。

将打开队列, 名称列表, 进程定义或队列管理器以用于后续 MQINQ 调用。

此选项对除分发列表以外的所有类型的对象都有效。如果 *ObjectQMgrName* 是队列管理器别名的名称, 那么此属性无效; 即使用于队列管理器别名判别的远程队列的本地定义中的 *RemoteQMgrName* 属性值是本地队列管理器的名称, 也是如此。

MQOO_SET

打开队列以设置属性。

打开队列以与后续 MQSET 调用配合使用。

此选项对除分发列表以外的所有类型的队列都有效。如果 *ObjectQMgrName* 是远程队列的本地定义的名称，那么此属性无效；即使用于队列管理器别名判别的远程队列的本地定义中的 *RemoteQMgrName* 属性值是本地队列管理器的名称，也是如此。

绑定选项: 当要打开的对象是集群队列时，以下选项适用；这些选项控制队列句柄与集群队列实例的绑定：

MQOO_BIND_ON_OPEN

当打开队列时，本地队列管理器将队列句柄绑定到目标队列的实例。因此，使用此句柄放入的所有消息都将发送到目标队列的同一实例，并通过同一路径发送。

此选项仅对于队列有效，并且仅影响集群队列。如果为不是集群队列的队列指定此选项，则会忽略此选项。

MQOO_BIND_NOT_FIXED

这将停止本地队列管理器将队列句柄绑定到目标队列的实例。因此，使用此句柄的连续 MQPUT 调用会将消息发送到目标队列的不同实例，或者发送到同一实例，但通过不同的路径。它还允许选择的实例稍后由本地队列管理器，远程队列管理器或消息通道代理程序 (MCA) 根据网络条件进行更改。

注: 需要交换系列消息以完成事务的客户机和服务器应用程序不得使用 MQOO_BIND_NOT_FIXED (或者当 *DefBind* 具有值 MQBND_BIND_NOT_FIXED 时使用 MQOO_BIND_AS_Q_DEF)，因为系列中的连续消息可能会发送到服务器应用程序的不同实例。

如果为集群队列指定了 MQOO_BROWSE 或其中一个 MQOO_INPUT_* 选项，那么将强制队列管理器选择集群队列的本地实例。因此，队列句柄的绑定是固定的，即使指定了 MQOO_BIND_NOT_FIXED 也是如此。

如果使用 MQOO_BIND_NOT_FIXED 指定了 MQOOO_INQUIRE，那么使用该句柄的连续 MQINQ 调用可能会查询集群队列的不同实例，尽管通常所有实例都具有相同的属性值。

MQOO_BIND_NOT_FIXED 仅对队列有效，并且仅影响集群队列。如果为不是集群队列的队列指定此选项，则会忽略此选项。

MQOO_BIND_ON_GROUP

允许应用程序请求将一组消息全部分配给同一目标实例。

此选项仅对于队列有效，并且仅影响集群队列。如果为不是集群队列的队列指定此选项，则会忽略此选项。

MQOO_BIND_AS_Q_DEF

本地队列管理器以 *DefBind* 队列属性定义的方式绑定队列句柄。此属性的值为 MQBND_BIND_ON_OPEN，MQBND_BIND_NOT_FIXED 或 MQBND_BIND_ON_GROUP。

当未指定 MQOO_BIND_ON_OPEN，MQOO_BIND_NOT_FIXED 或 MQOO_BIND_ON_GROUP 时，MQOO_BIND_AS_Q_DEF 是缺省值。

MQOO_BIND_AS_Q_DEF 辅助程序文档。不打算将此选项与其他两个绑定选项中的任何一个一起使用，但由于其值为零，因此无法检测到使用情况。

上下文选项: 以下选项控制消息上下文的处理：

MQOO_SAVE_ALL_CONTEXT

上下文信息与此队列句柄相关联。此信息是从使用此句柄检索的任何消息的上下文中设置的。有关消息上下文的更多信息，请参阅 [消息上下文](#) 和 [控制上下文信息](#)。

可以将此上下文信息传递到消息，然后使用 MQPUT 或 MQPUT1 调用将该消息放入队列中。请参阅第 433 页的『MQPMO-Put-message 选项』中描述的 MQPMO_PASS_IDENTITY_CONTEXT 和 MQPMO_PASS_ALL_CONTEXT 选项。

在成功检索消息之前，无法将上下文传递到正在放入队列中的消息。

使用其中一个 MQGMO_BROWSE_* 浏览选项检索的消息未保存其上下文信息 (尽管 *MsgDesc* 参数中的上下文字段是在浏览后设置的)。

此选项仅对本地队列，别名队列和模型队列有效；对于非队列的远程队列，分发列表和对象无效。必须指定其中一个 MQOO_INPUT_* 选项。

MQOO_PASS_IDENTITY_CONTEXT

这允许在将消息放入队列时在 *PutMsgOpts* 参数中指定 `MQPMO_PASS_IDENTITY_CONTEXT` 选项; 这将为消息提供来自使用 `MQOO_SAVE_ALL_CONTEXT` 选项打开的输入队列的身份上下文信息。有关消息上下文的更多信息, 请参阅 [消息上下文](#) 和 [控制上下文信息](#)。

必须指定 `MQOO_OUTPUT` 选项。

此选项对所有类型的队列 (包括分发列表) 都有效。

MQOO_PASS_ALL_CONTEXT

这允许在将消息放入队列时在 *PutMsgOpts* 参数中指定 `MQPMO_PASS_ALL_CONTEXT` 选项; 这将为消息提供来自使用 `MQOO_SAVE_ALL_CONTEXT` 选项打开的输入队列的身份和源上下文信息。有关消息上下文的更多信息, 请参阅 [消息上下文](#) 和 [控制上下文信息](#)。

此选项暗示 `MQOO_PASS_IDENTITY_CONTEXT`, 因此无需指定。必须指定 `MQOO_OUTPUT` 选项。

此选项对所有类型的队列 (包括分发列表) 都有效。

MQOO_SET_IDENTITY_CONTEXT

这允许在将消息放入队列时在 *PutMsgOpts* 参数中指定 `MQPMO_SET_IDENTITY_CONTEXT` 选项; 这将为消息提供包含在 `MQPUT` 或 `MQPUT1` 调用上指定的 *MsgDesc* 参数中的身份上下文信息。有关消息上下文的更多信息, 请参阅 [消息上下文](#) 和 [控制上下文信息](#)。

此选项暗示 `MQOO_PASS_IDENTITY_CONTEXT`, 因此无需指定。必须指定 `MQOO_OUTPUT` 选项。

此选项对所有类型的队列 (包括分发列表) 都有效。

MQOO_SET_ALL_CONTEXT

这允许在将消息放入队列时在 *PutMsgOpts* 参数中指定 `MQPMO_SET_ALL_CONTEXT` 选项; 这将向消息提供 `MQPUT` 或 `MQPUT1` 调用上指定的 *MsgDesc* 参数中包含的身份和源上下文信息。有关消息上下文的更多信息, 请参阅 [消息上下文](#) 和 [控制上下文信息](#)。

此选项意味着以下选项, 因此无需指定这些选项:

- `MQOO_PASS_IDENTITY_CONTEXT`
- `MQOO_PASS_ALL_CONTEXT`
- `MQOO_SET_IDENTITY_CONTEXT`

必须指定 `MQOO_OUTPUT` 选项。

此选项对所有类型的队列 (包括分发列表) 都有效。

预读选项:

使用 `MQOO_READ_AHEAD` 调用 `MQOPEN` 时, 仅当满足特定条件时, WebSphere MQ 客户机才会启用预读。这些条件包括:

- 客户机和远程队列管理器都必须处于 WebSphere MQ V7 或更高版本。
- 必须针对线程 WebSphere MQ MQI 客户机库编译和链接客户机应用程序。
- 客户机通道必须使用的是 TCP/IP 协议
- 该通道必须在客户机和服务器通道定义中具有非零 `SharingConversations (SHARECNV)` 设置。

以下选项控制在应用程序请求非持久消息之前是否将这些消息发送到客户机。以下说明适用于预读选项:

- 只能指定其中一个选项。
- 这些选项仅对本地队列, 别名队列和模型队列有效。它们对于远程队列, 分发列表, 主题或队列管理器无效。
- 仅当还指定了 `MQOOO_BROWSE`, `MQOO_INPUT_SHARED` 和 `MQOO_INPUT_EXCLUSIVE` 之一时, 这些选项才适用, 尽管使用 `MQOOO_INQUIRE` 或 `MQOO_SET` 指定这些选项不是错误。
- 如果应用程序未作为 IBM WebSphere MQ 客户机运行, 那么将忽略这些选项。

MQOO_NO_READ_AHEAD

在应用程序请求非持久消息之前, 不会向客户机发送这些消息。

MQOO_READ_AHEAD

在应用程序请求非持久消息之前，会将这些消息发送到客户机。

MQOO_READ_AHEAD_AS_Q_DEF

预读行为由正在打开的队列的缺省预读属性确定。这是缺省值。

其他选项: 以下选项控制授权检查，队列管理器停顿时发生的情况，是否解析本地队列名称以及多点广播:

MQOO_ALTERNATE_USER_AUTHORITY

ObjDesc 参数中的 *AlternateUserId* 字段包含用于验证此 MQOPEN 调用的用户标识。仅当此 *AlternateUserId* 有权使用指定的访问选项打开对象时，该调用才能成功，而不管运行应用程序的用户标识是否有权执行此操作。这不适用于指定的任何上下文选项，但是，将始终根据运行应用程序的用户标识来检查这些上下文选项。

此选项对所有类型的对象都有效。

MQOO_FAIL_IF QUIESCING

如果队列管理器处于停顿状态，那么 MQOPEN 调用将失败。

在 z/OS 上，对于 CICS 或 IMS 应用程序，如果连接处于停顿状态，那么此选项还会强制 MQOPEN 调用失败。

此选项对所有类型的对象都有效。

有关客户机通道的信息，请参阅 [IBM WebSphere MQ MQI 客户机概述](#)。

MQOO_RESOLVE_LOCAL_Q

使用打开的本地队列的名称填充 MQOD 结构中的 ResolvedQName。同样，ResolvedQMgr 名称中填充了托管本地队列的本地队列管理器的名称。如果 MQOD 结构低于 V 3，那么将忽略 MQOO_RESOLVE_LOCAL_Q 而不返回任何错误。

当打开本地队列，别名队列或模型队列时，将始终返回本地队列，但如果在未使用 MQOO_RESOLVE_LOCAL_Q 选项的情况下打开远程队列或非本地集群队列，那么情况并非如此；ResolvedQName 和 ResolvedQMgr 名称将使用远程队列定义中找到的 RemoteQName 和 RemoteQMgr 名称填充，或者与所选远程集群队列类似。

如果在打开 (例如) 远程队列时指定 MQOO_RESOLVE_LOCAL_Q，那么 ResolvedQName 是将消息放入的传输队列。ResolvedQMgr 名称由托管传输队列的本地队列管理器的名称填充。

如果您有权在队列上进行浏览，输入或输出，那么您具有在 MQOPEN 调用上指定此标志的必需权限。不需要特殊权限。

此选项仅对队列和队列管理器有效。

MQOO_RESOLVE_LOCAL_TOPIC

使用打开的管理主题的名称填充 MQOD 结构中的 ResolvedQName。

MQOO_NO_多点广播

不使用多点广播发送发布消息。

此选项仅对 MQOO_OUTPUT 选项有效。如果未指定 MQOO_OUTPUT，那么 MQOPEN 将返回 MQRC_OPTIONS_ERROR。

此选项仅对主题有效。

Hobj

类型:MQHOBJ-输出

此句柄表示已建立的对该对象的访问权。必须在对该对象进行操作的后续 IBM WebSphere MQ 调用上指定此参数。当发出 MQCLOSE 调用时，或者当定义句柄作用域的处理单元终止时，它将停止有效。

返回的对象句柄的作用域与调用上指定的连接句柄的作用域相同。有关句柄作用域的信息，请参阅 [MQCONN-Hconn 参数](#)。

CompCode

类型: MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_WARNING

警告（部分完成）。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING：

MQRC_MULTIPLE_原因

(2136, X'858') 返回了多个原因码。

如果 *CompCode* 为 MQCC_FAILED：

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ALIAS_BASE_Q_TYPE_ERROR

(2001, X'7D1') 别名基本队列不是有效类型。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CF_NOT_AVAILABLE

(2345, X'929') 耦合设施不可用。

MQRC_CF_STRUC_AUTH_FAILED

(2348, X'92C') 耦合设施结构授权检查失败。

MQRC_CF_STRUC_ERROR

(2349, X'92D') 耦合设施结构无效。

MQRC_CF_STRUC_FAILED

(2373, X'945') 耦合设施结构失败。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') 正在使用耦合设施结构 list-header。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CLUSTER_EXIT_ERROR

(2266, X'8DA') 集群工作负载出口失败。

MQRC_CLUSTER_PUT_禁止
(2268, X'8DC') 对集群中的所有队列禁止 Put 调用。

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') 集群名称解析失败。

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') 集群资源错误。

MQRC_CONNECTION_BROKEN
(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') 未授权连接。

MQRC_CONNECTION_QUIESCING
(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING
(2203, X'89B') 连接正在关闭。

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') Db2 子系统不可用。

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896') 缺省传输队列不是本地传输队列。

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897') 缺省传输队列使用错误。

MQRC_DYNAMIC_Q_NAME_ERROR
(2011, X'7DB') 动态队列的名称无效。

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') 没有更多可用的句柄。

MQRC_HCONN_ERROR
(2018, X'7E2') 连接句柄无效。

MQRC_HOBJ_ERROR
(2019, X'7E3') 对象句柄无效。

MQRC_MULTIPLE_原因
(2136, X'858') 返回了多个原因码。

MQRC_NAME_IN_USE
(2201, X'899') 名称正在使用中。

MQRC_NAME_NOT_VALID_FOR_TYPE
(2194, X'892') 对象名对于对象类型无效。

MQRC_NOT_AUTHORIZED
(2035, X'7F3') 未获得访问授权。

MQRC_OBJECT_ALREADY_EXISTS
(2100, X'834') 对象存在。

MQRC_OBJECT_DAMAGED
(2101, X'835') 对象已损坏。

MQRC_OBJECT_IN_USE
(2042, X'7FA') 对象已打开，但有冲突的选项。

MQRC_OBJECT_LEVEL_不兼容
(2360, X'938') 对象级别不兼容。

MQRC_OBJECT_NAME_ERROR
(2152, X'868') 对象名无效。

MQRC_OBJECT_NOT_UNIQUE
(2343, X'927') 对象不唯一。

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869') 对象队列管理器名称无效。

MQRC_OBJECT_RECORDS_ERROR

(2155, X'86B') 对象记录无效。

MQRC_OBJECT_STRING_ERROR

(2441, X'0989 ') Objectstring 字段无效

MQRC_OBJECT_TYPE_ERROR

(2043, X'7FB') 对象类型无效。

MQRC_OD_ERROR

(2044, X'7FC') 对象描述符结构无效。

MQRC_OPTION_NOT_VALID_FOR_TYPE

(2045, X'7FD') 选项对于对象类型无效。

MQRC_OPTIONS_ERROR

(2046, X'7FE') 选项无效或不一致。

MQRC_PAGESET_ERROR

(2193, X'891 ') 访问页集数据集时出错。

MQRC_PAGESET_FULL

(2192, X'890 ') 外部存储介质已满。

MQRC_Q_DELETED

(2052, X'804 ') 队列已删除。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING

(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_Q_TYPE_ERROR

(2057, X'809 ') 队列类型无效。

MQRC_RECS_PRESENT_ERROR

(2154, X'86A') 存在的记录数无效。

MQRC_REMOTE_Q_NAME_ERROR

(2184, X'888 ') 远程队列名无效。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_RESPONSE_RECORDS_ERROR

(2156, X'86C') 响应记录无效。

MQRC_SECURITY_ERROR

(2063, X'80F') 发生了安全性错误。

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') 发出了 MQOPEN, MQPUT1 或 MQSUB 调用, 但指定了包含语法错误的选择字符串。

MQRC_STOPPED_BY_CLUSTER_EXIT

(2188, X'88C') 集群工作负载出口拒绝调用。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') 外部存储介质已满。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口程序禁止调用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

MQRC_UNKNOWN_ALIAS_BASE_Q

(2082, X'822 ') 未知别名基本队列。

MQRC_UNKNOWN_DEF_XMIT_Q

(2197, X'895 ') 未知缺省传输队列。

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825 ') 未知对象名。

MQRC_UNKNOWN_OBJECT_Q_MGR

(2086, X'826 ') 未知对象队列管理器。

MQRC_UNKNOWN_REMOTE_Q_MGR

(2087, X'827 ') 未知远程队列管理器。

MQRC_UNKNOWN_XMIT_Q

(2196, X'894 ') 未知传输队列。

MQRC_不法_cf_level

(2366, X'93E') 耦合设施结构级别错误。

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') 传输队列不是本地的。

MQRC_XMIT_Q_USAGE_ERROR

(2092, X'82C') 传输队列使用错误。

有关这些代码的详细信息，请参阅：

- 所有其他 IBM WebSphere MQ 平台 (z/OS 除外) 的 [原因码](#)。

一般使用说明

1. 打开的对象是下列其中一项：

- 要执行以下操作的队列：
 - 获取或浏览消息 (使用 MQGET 调用)
 - 放置消息 (使用 MQPUT 调用)
 - 查询队列的属性 (使用 MQINQ 调用)
 - 设置队列的属性 (使用 MQSET 调用)

如果指定的队列是模型队列，那么将创建动态本地队列。请参阅 [第 642 页的『MQOPEN-打开对象』](#) 中描述的 *ObjDesc* 参数。

分发列表是包含队列列表的特殊类型的队列对象。可以打开它来放置消息，但不能获取或浏览消息，也不能查询或设置属性。请参阅 [用法说明 8](#) 以获取更多详细信息。

具有 QSGDISP(GROUP) 的队列是无法与 MQOPEN 或 MQPUT1 调用配合使用的特殊类型的队列定义。

- 用于查询列表中队列的名称的名称列表 (使用 MQINQ 调用)。
 - 用于查询进程属性的进程定义 (使用 MQINQ 调用)。
 - 用于查询本地队列管理器属性的队列管理器 (使用 MQINQ 调用)。
 - 用于发布消息的主题 (使用 MQPUT 调用)
2. 应用程序可以多次打开同一对象。对于每个打开的对象，将返回不同的对象句柄。返回的每个句柄都可用于执行相应打开的功能。
3. 如果要打开的对象是除集群队列以外的队列，那么本地队列管理器中的所有名称解析都在 MQOPEN 调用时进行。这可能包括：
- 将远程队列的本地定义的名称解析为远程队列管理器的名称，以及该队列在远程队列管理器上的已知名称
 - 将远程队列管理器名称解析为本地传输队列的名称

- (仅限 z/OS) 将远程队列管理器名称解析为 IGQ 代理程序所使用的共享传输队列的名称 (仅当本地和远程队列管理器属于同一队列共享组时才适用)
- 基本队列或主题对象的名称的别名解析。

但是, 请注意, 句柄的后续 MQINQ 或 MQSET 调用仅与已打开的名称相关, 而不是与发生名称解析后生成的对象相关。例如, 如果打开的对象是别名, 那么 MQINQ 调用返回的属性是别名的属性, 而不是基本队列的属性或别名解析为的主题对象。

如果要打开的对象是集群队列, 那么可以在 MQOPEN 调用时进行名称解析, 也可以延迟到以后。发生解析的点由 MQOPEN 调用上指定的 MQOO_BIND_* 选项控制:

- MQOO_BIND_ON_OPEN
- MQOO_BIND_NOT_FIXED
- MQOO_BIND_AS_Q_DEF
- MQOO_BIND_ON_GROUP

有关集群队列的名称解析的更多信息, 请参阅 [名称解析](#)。

4. 带有 MQOO_BROWSE 选项的 MQOPEN 调用建立一个浏览游标, 用于指定对象句柄和其中一个浏览选项的 MQGET 调用。这允许在不改变其内容的情况下扫描队列。可以使用 MQGMO_MSG_UNDER_CURSOR 选项从队列中除去通过浏览找到的消息。

通过对同一队列发出多个 MQOPEN 请求, 可以对单个应用程序激活多个浏览游标。

5. 由触发器监视器启动的应用程序将在应用程序启动时传递与该应用程序相关联的队列的名称。可以在 *ObjDesc* 参数中指定此队列名称以打开队列。有关更多详细信息, 请参阅 [第 528 页的『MQTMC2 - 触发器消息 2 \(字符格式\)』](#)。
6. 在 IBM i 上, 以兼容性方式运行的应用程序将通过应用程序发出的第一个 MQOPEN 调用自动连接到队列管理器 (如果应用程序尚未使用 MQCONN 调用连接到队列管理器)。

未以兼容性方式运行的应用程序必须先发出 MQCONN 或 MQCONNX 调用以显式连接到队列管理器, 然后再使用 MQOPEN 调用来打开对象。

预读选项

使用 MQOO_READ_AHEAD 调用 MQOPEN 时, 仅当满足特定条件时, WebSphere MQ 客户机才会启用预读。这些条件包括:

- 客户机和远程队列管理器都必须处于 WebSphere MQ V7 或更高版本。
- 必须针对线程 WebSphere MQ MQI 客户机库编译和链接客户机应用程序。
- 客户机通道必须使用的是 TCP/IP 协议
- 该通道必须在客户机和服务器通道定义中具有非零 SharingConversations (SHARECNV) 设置。

以下说明适用于预读选项的使用。

1. 仅当同时指定了 MQOO_BROWSE, MQOO_INPUT_SHARED 和 MQO_INPUT_EXCLUSIVE 选项中的一个且仅一个时, 预读选项才适用。如果使用 MQOO_INQUIRE 或 MQOO_SET 选项指定了预读选项, 那么不会抛出错误。
2. 如果第一个 MQGET 调用上使用的选项不支持用于预读, 那么在请求时不会启用预读。此外, 当客户机连接到不支持预读的队列管理器时, 将禁用预读。
3. 如果应用程序未作为 IBM WebSphere MQ 客户机运行, 那么将忽略预读选项。

集群队列

以下说明适用于集群队列的使用。

1. 首次打开集群队列时, 如果本地队列管理器不是完整存储库队列管理器, 那么本地队列管理器将从完整存储库队列管理器获取有关集群队列的信息。当网络繁忙时, 本地队列管理器可能需要几秒钟才能从存储库队列管理器接收所需信息。因此, 发出 MQOPEN 调用的应用程序可能需要等待最多 10 秒才能从 MQOPEN 调用返回控制权。如果本地队列管理器在此时间内未收到有关集群队列的所需信息, 那么调用将失败, 原因码为 MQRC_CLUSTER_RESOLUTION_ERROR。

2. 当打开集群队列并且集群中有多个队列实例时，打开的实例取决于 MQOPEN 调用上指定的选项：

- 如果指定的选项包含以下任何选项：
 - MQOO_BROWSE
 - MQOO_INPUT_AS_Q_DEF
 - MQOO_INPUT_EXCLUSIVE
 - MQOO_INPUT_SHARED
 - MQOO_SET

打开的集群队列的实例必须是本地实例。如果没有队列的本地实例，那么 MQOPEN 调用将失败。

- 如果指定的选项不包含先前描述的任何选项，但包含以下一个或两个选项：
 - MQOO_INQUIRE
 - MQOO_OUTPUT

如果存在本地实例，那么打开的实例是本地实例，否则是远程实例 (如果使用 CLWLULUSEQ 缺省值)。但是，队列管理器选择的实例可以由集群工作负载出口 (如果有) 变更。

3. 如果存在该队列的预订，但该预订未被完整存储库确认，那么该对象在集群中不存在，并且调用失败，原因码为 MQRC_OBJECT_NAME。

有关集群队列的更多信息，请参阅 [集群队列](#)。

分发列表

以下说明适用于分发列表的使用。

在以下环境中支持分发列表：AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 IBM WebSphere MQ MQI 客户机。

1. 打开分发列表时，必须按如下所示设置 MQOD 结构中的字段：

- *Version* 必须为 MQOD_VERSION_2 或更高版本。
- *ObjectType* 必须是 MQOT_Q。
- *ObjectName* 必须为空白或空字符串。
- *ObjectQMgrName* 必须为空白或空字符串。
- *RecsPresent* 必须大于零。
- 其中一个 *ObjectRecOffset* 和 *ObjectRecPtr* 必须为零，另一个非零。
- 不能有多个 *ResponseRecOffset* 和 *ResponseRecPtr* 非零。
- 必须有 *RecsPresent* 个对象记录，由 *ObjectRecOffset* 或 *ObjectRecPtr* 寻址。必须将对象记录设置为要打开的目标队列的名称。
- 如果 *ResponseRecOffset* 和 *ResponseRecPtr* 中的一个非零，那么必须存在 *RecsPresent* 个响应记录。如果调用完成且原因码为 MQRC_MULTIPLE_REASON，那么队列管理器会设置这些参数。

通过确保 *RecsPresent* 为零，还可以使用 version-2 MQOD 来打开不在分发列表中的单个队列。

2. 只有下列打开选项在 *Options* 参数中有效：

- MQOO_OUTPUT
- MQOO_PASS_*_CONTEXT
- MQOO_SET_*_CONTEXT
- MQOO_ALTERNATE_USER_AUTHORITY
- MQOO_FAIL_IF_QUIESCING

3. 分发列表中的目标队列可以是本地队列，别名队列或远程队列，但它们不能是模型队列。如果指定了模型队列，那么该队列无法打开，原因码为 MQRC_Q_TYPE_ERROR。但是，这不会阻止成功打开列表中的其他队列。

4. 完成代码和原因码参数设置如下：

- 如果分发列表中队列的打开操作全部成功或以相同方式失败，那么将设置完成代码和原因码参数以描述公共结果。在这种情况下，未设置 MQRR 响应记录 (如果由应用程序提供)。

例如，如果每次打开都成功，那么完成代码将设置为 MQCC_OK，原因码将设置为 MQRC_NONE；如果每次打开都失败，因为不存在任何队列，那么参数将设置为 MQCC_FAILED 和 MQRC_UNKNOWN_OBJECT_NAME。
- 如果分发列表中队列的打开操作并非全部成功或以相同方式失败：
 - 如果至少有一个打开成功，那么完成代码参数将设置为 MQCC_WARNING，如果所有操作都失败，那么将设置为 MQCC_FAILED。
 - 原因码参数设置为 MQRC_MULTIPLE_REASON。
 - 响应记录 (如果由应用程序提供) 将设置为分发列表中队列的各个完成代码和原因码。
- 5. 成功打开分发列表后，调用返回的句柄 *Hobj* 可用于后续 MQPUT 调用，以将消息放入分发列表中的队列，以及 MQCLOSE 调用以放弃对分发列表的访问权。分发列表的唯一有效关闭选项是 MQCO_NONE。MQPUT1 调用也可用于将消息放入分发列表；定义列表中的队列的 MQOD 结构被指定为该调用上的参数。
- 6. 当检查应用程序是否已超过允许的最大句柄数时，分发列表中的每个成功打开的目标都将计为一个单独的句柄 (请参阅 *MaxHandles* queue-manager 属性)。即使当分发列表中的两个或更多目标解析为同一物理队列时，也是如此。如果针对分发列表的 MQOPEN 或 MQPUT1 调用将导致应用程序正在使用的句柄数超过 *MaxHandles*，那么调用将失败，原因码为 MQRC_HANDLE_NOT_AVAILABLE。
- 7. 成功打开的每个目标都将其 *OpenOutputCount* 属性的值递增 1。如果分发列表中的两个或多个目标解析为同一物理队列，那么该队列的 *OpenOutputCount* 属性将按分发列表中解析为该队列的目标数递增。
- 8. 如果对队列定义的任何更改 (例如，解析路径中的更改) 会导致在单独打开队列时句柄变为无效，那么这些更改不会导致分发列表句柄变为无效。但是，在后续 MQPUT 调用上使用分发列表句柄时，会导致该特定队列发生故障。
- 9. 分发列表只能包含一个目标。

远程队列

以下说明适用于远程队列的使用。

可以通过此调用的 *ObjDesc* 参数中的两种方法之一来指定远程队列。

- 通过为 *ObjectName* 指定远程队列的本地定义的名称。在这种情况下，*ObjectQMGrName* 引用本地队列管理器，并且可以将其指定为空白或 (在 C 编程语言中) 空字符串。

本地队列管理器执行的安全性验证将验证用户是否有权打开远程队列的本地定义。
- 通过为 *ObjectName* 指定远程队列管理器已知的远程队列名称。在这种情况下，*ObjectQMGrName* 是远程队列管理器的名称。

由本地队列管理器执行的安全性验证将验证用户是否有权将消息发送到由名称解析过程产生的传输队列。

在任何一种情况下：

- 本地队列管理器不会向远程队列管理器发送任何消息，以检查用户是否有权将消息放入队列中。
- 当消息到达远程队列管理器时，远程队列管理器可能会拒绝该消息，因为发起该消息的用户未经授权。

有关更多信息，请参阅第 414 页的『MQOD-对象描述符』中描述的 *ObjectName* 和 *ObjectQMGrName* 字段。

对象

安全性

以下说明涉及使用 MQOPEN 的安全方面。

队列管理器在发出 MQOPEN 调用时执行安全性检查，以验证在允许访问之前，运行应用程序的用户标识是否具有相应的权限级别。将对要打开的对象名称进行权限检查，而不是对在解析名称后生成的名称进行权限检查。

如果要打开的对象是指向主题对象的别名队列，那么队列管理器会先对别名队列名称执行安全性检查，然后再对主题执行安全性检查，就像直接使用了主题对象一样。

如果要打开的对象是主题对象 (无论是单独使用 *ObjectName* 还是使用 *ObjectString* (具有或不具有基本 *ObjectName*))，那么队列管理器将通过使用从 *ObjectName* 中指定的主题对象内获取的结果主题字符串来执行安全性检查，如果需要将其与 *ObjectString* 中提供的主题对象并置，然后在主题树中的该点或该点以上找到最接近的主题对象以执行安全性检查。这可能与 *ObjectName* 中指定的主题对象不同。

如果要打开的对象是模型队列，那么队列管理器将对模型队列的名称和创建的动态队列的名称执行完全安全性检查。如果随后显式打开生成的动态队列，那么将针对动态队列的名称执行进一步的资源安全性检查。

属性

以下注释与属性相关。

当应用程序打开对象时，可以更改该对象的属性。在许多情况下，应用程序不会注意到这一点，但对于某些属性，队列管理器会将句柄标记为不再有效。这些属性为：

- 影响对象的名称解析的任何属性。这将适用，而不考虑所使用的打开选项，包括以下内容：
 - 对已打开的别名队列的 *BaseQName* 属性的更改。
 - 对已打开的别名队列的 *TargetType* 属性的更改。
 - 对 *RemoteQName* 或 *RemoteQMGrName* 队列属性的更改，对于为此队列打开的任何句柄，或者对于通过此定义解析为队列管理器别名的队列。
 - 导致远程队列的当前打开句柄解析为另一传输队列或根本无法解析为一个传输队列的任何更改。例如，这可以包括：
 - 对远程队列的本地定义的 *XmitQName* 属性的更改，无论该定义是用于队列还是用于队列管理器别名。
 - (仅适用于 z/OS) 对 *IntraGroupQueuing* 队列管理器属性值的更改，或对共享传输队列 (SYSTEM.QSG.TRANSMIT.QUEUE) 由 IGQ 代理程序使用。
- 有一个例外：创建新的传输队列。如果在打开句柄时存在此队列，但改为解析为缺省传输队列，那么将解析到此队列的句柄不会变为无效。
- 对 *DefXmitQName* 队列管理器属性的更改。在这种情况下，解析为先前指定的队列 (仅因为它是缺省传输队列而解析为该队列) 的所有打开句柄都被标记为无效。由于其他原因解析到此队列的句柄不受影响。
- *Shareability* 队列属性，如果有两个或多个句柄当前正在为此队列提供 MQOO_INPUT_SHARED 访问权，或者为解析到此队列的队列提供 MQOO_INPUT_SHARED 访问权。如果是这样，那么针对此队列或解析到此队列的队列打开的所有句柄都将标记为无效，而不考虑打开的选项。

在 z/OS 上，如果一个或多个句柄当前正在提供对队列的 MQOO_INPUT_SHARED 或 MQOO_INPUT_EXCLUSIVE 访问权，那么先前描述的句柄将标记为无效。

- *Usage* 队列属性，针对为此队列打开的所有句柄，或者针对解析到此队列的队列，而不考虑打开的选项。

当句柄标记为无效时，使用此句柄的所有后续调用 (MQCLOSE 除外) 都将失败，原因码为 MQRC_OBJECT_CHANGED。应用程序必须发出 MQCLOSE 调用 (使用原始句柄)，然后重新打开队列。根据应用程序逻辑的要求，仍可以落实或回退来自先前成功调用的旧句柄的任何未落实更新。

如果更改属性导致发生此情况，请使用特殊强制版本的调用。

C 调用

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
        &Reason);
```


按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

按如下所示声明参数:

```
** 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.
```

PL/I 调用

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

按如下所示声明参数:

```
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 */
```

高级汇编程序调用

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

按如下所示声明参数:

```
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
```

Visual Basic 调用

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

按如下所示声明参数:

```
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-放置消息

MQPUT 调用将消息放入队列或分发列表中, 或放入主题中。队列, 分发列表或主题必须已打开。

语法

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。

在针对 CICS 应用程序的 z/OS 上, 以及针对以兼容性方式运行的应用程序的 IBM i 上, 可以省略 MQCONN 调用, 并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

Hobj

类型:MQHOBJ-输入

此句柄表示将消息添加到的队列或将消息发布到的主题。*Hobj* 的值由指定了 MQOO_OUTPUT 选项的先前 MQOPEN 调用返回。

MsgDesc

类型:MQMD-输入/输出

此结构描述要发送的消息的属性, 并在 put 请求完成后接收有关消息的信息。请参阅第 364 页的『MQMD - 消息描述符』, 以了解详细信息。

如果应用程序提供了 version-1 MQMD, 那么可以使用 MQMDE 结构作为消息数据的前缀, 以指定 version-2 MQMD 中存在但 version-1 中不存在的字段的值。MQMD 中的格式字段必须设置为 MQFMT_MD_EXTENSION, 以指示存在 MQMDE。请参阅第 406 页的『MQMDE-消息描述符扩展』以获取更多详细信息。

如果在 MQPMO 结构的 *OriginalMsgHandle* 或 *NewMsgHandle* 字段中提供了有效的消息句柄, 那么应用程序不需要提供 MQMD 结构。如果在其中一个字段中未提供任何内容, 那么将从与消息句柄关联的描述符中获取消息描述符。

如果您使用或计划使用 API 出口, 那么建议您显式提供 MQMD 结构, 而不要使用与消息句柄关联的消息描述符。这是因为与 MQPUT 或 MQPUT1 调用关联的 API 出口无法确定队列管理器使用哪些 MQMD 值来完成 MQPUT 或 MQPUT1 请求。

PutMsg 选项

类型:MQPMO-输入/输出

请参阅第 433 页的『MQPMO-Put-message 选项』, 以了解详细信息。

BufferLength

类型:MLONG-输入

Buffer 中消息的长度。零有效, 指示消息不包含应用程序数据。*BufferLength* 的上限取决于各种因素:

- 如果目标是本地队列或解析为本地队列，那么上限取决于是否：
 - 本地队列管理器支持分段。
 - 发送应用程序指定允许队列管理器对消息进行分段的标志。此标志是 `MQMF_SEGMENTATION_ALLOWED`，可以在 `version-2 MQMD` 中指定，也可以在与 `version-1 MQMD` 配合使用的 `MQMDE` 中指定。

如果满足这两个条件，那么 `BufferLength` 不能超过 `999 999 999` 减去 `MQMD` 中 `Offset` 字段的值。因此，可放入的最长逻辑消息为 `999 999 999` 字节 (当 `Offset` 为零时)。但是，运行应用程序的操作系统或环境施加的资源约束可能会导致下限。

如果不满足上述其中一个或两个条件，那么 `BufferLength` 不能超过队列的 `MaxMsgLength` 属性和队列管理器的 `MaxMsgLength` 属性中较小的属性。

- 如果目标是远程队列或解析为远程队列，那么本地队列的条件适用，但必须在每个队列管理器上传递消息才能到达目标队列；尤其是：
 1. 用于在本地队列管理器中临时存储消息的本地传输队列
 2. 用于在本地队列管理器与目标队列管理器之间的路由上的队列管理器上存储消息的中间传输队列 (如果有)
 3. 目标队列管理器上的目标队列

因此，可以放入的最长消息由这些队列和队列管理器中限制最大的队列和队列管理器管理。

当消息位于传输队列上时，附加信息与消息数据一起存在，这将减少可携带的应用程序数据量。在此情况下，在确定 `BufferLength` 的限制时，请从传输队列的 `MaxMsgLength` 值中减去 `MQ_MSG_HEADER_LENGTH` 字节。

注：放入消息时，只能同步诊断是否符合条件 1 (原因码为 `MQRC_MSG_TOO_BIG_FOR_Q` 或 `MQRC_MSG_TOO_BIG_FOR_Q_MGR`)。如果不满足条件 2 或 3，那么会将消息重定向到中间队列管理器或目标队列管理器上的死信 (未传递消息) 队列。如果发生这种情况，那么如果发件人请求了报告消息，那么将生成报告消息。

buffer

类型: `MQBYTEExBuffer` 长度-输入

这是一个缓冲区，其中包含要发送的应用程序数据。缓冲区必须在适合于消息中数据的性质的边界上对齐。4 字节对齐适用于大多数消息 (包括包含 `WebSphere MQ` 头结构的消息)，但某些消息可能需要更严格的对齐。例如，包含 64 位二进制整数的消息可能需要 8 字节对齐。

如果 `Buffer` 包含字符或数字数据，请将 `MsgDesc` 参数中的 `CodedCharSetId` 和 `Encoding` 字段设置为适合于数据的值；这使消息接收方能够将数据 (如果需要) 转换为接收方使用的字符集和编码。

注：`MQPUT` 调用上的所有其他参数必须采用本地队列管理器的字符集和编码 (由 `CodedCharSetId` `queue-manager` 属性和 `MQENC_NATIVE` 提供)。

在 C 编程语言中，该参数被声明为指向 `void` 的指针；任何类型的数据的地址都可以被指定为该参数。

如果 `BufferLength` 参数为零，那么不会引用 `Buffer`；在此情况下，以 C 或 `System/390` 汇编程序编写的程序传递的参数地址可以为空。

CompCode

类型: `MLONG` - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: `MLONG` - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') 消息组未完成。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 逻辑消息未完成。

MQRC_INCONSISTENT_PERSISTENCE

(2185, X'889') 持久性规范不一致。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 工作单元规范不一致。

MQRC_MULTIPLE_原因

(2136, X'858') 返回了多个原因码。

MQRC_PRIORITY_EXCEEDS_MAXIMUM

(2049, X'801') 消息优先级超过支持的最大值。

MQRC_UNKNOWN_REPORT_OPTION

(2104, X'838') 无法识别消息描述符中的报告选项。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ALIAS_TARGTYPE_CHANGED

(2480, X'09B0') 预订目标类型已从队列更改为主题对象。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BACKED_OUT

(2003, X'7D3') 工作单元已回退。

MQRC_BUFFER_ERROR

(2004, X'7D4') 缓冲区参数无效。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CALL_中断

(2549, X'9F5') MQPUT 或 MQCMIT 已中断, 重新连接处理无法重新建立明确的结果。

MQRC_CF_STRUC_FAILED

(2373, X'945') 耦合设施结构失败。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CFGR_ERROR

(2416, X'970') 消息数据中的 PCF 组参数结构 MQCFGR 无效。

MQRC_CFH_ERROR
(2235, X'8BB') PCF 头结构无效。

MQRC_CFIF_ERROR
(2414, X'96E') 消息数据中的 PCF 整数过滤器参数结构无效。

MQRC_CFIL_ERROR
(2236, X'8BC') PCF 整数列表参数结构或 PCIF*64 整数列表参数结构无效。

MQRC_CFIN_ERROR
(2237, X'8BD') PCF 整数参数结构或 PCIF*64 整数参数结构无效。

MQRC_CFSF_ERROR
(2415, X'96F') 消息数据中的 PCF 字符串过滤器参数结构无效。

MQRC_CFSL_ERROR
(2238, X'8BE') PCF 字符串列表参数结构无效。

MQRC_CFST_ERROR
(2239, X'8BF') PCF 字符串参数结构无效。

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') 集群工作负载出口失败。

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') 集群名称解析失败。

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') 集群资源错误。

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') COD 报告选项对于 XCF 队列无效。

MQRC_CONNECTION_BROKEN
(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') 未授权连接。

MQRC_CONNECTION QUIESCING
(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING
(2203, X'89B') 连接正在关闭。

MQRC_CONTENT_ERROR
2554 (X'09FA) 无法解析消息内容以确定是否应使用扩展消息选择器将消息传递给订户。

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831 ') 引用的队列句柄不保存上下文。

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') 上下文不可用于引用的队列句柄。

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') 数据长度参数无效。

MQRC_DH_ERROR
(2135, X'857 ') 分发头结构无效。

MQRC_DLH_ERROR
(2141, X'85D') 死信头结构无效。

MQRC_EPH_ERROR
(2420, X'974 ') 嵌入式 PCF 结构无效。

MQRC_EXPIRY_ERROR
(2013 年, X'7DD') 到期时间无效。

MQRC_FEEDBACK_ERROR
(2014, X'7DE') 反馈代码无效。

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') 全局工作单元冲突。

MQRC_GROUP_ID_ERROR
(2258, X'8D2') 组标识无效。

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') 用于全局工作单元的句柄。

MQRC_HCONN_ERROR
(2018, X'7E2') 连接句柄无效。

MQRC_HEADER_ERROR
(2142, X'85E') MQ 头结构无效。

MQRC_HOBJ_ERROR
(2019, X'7E3') 对象句柄无效。

MQRC_IIH_ERROR
(2148, X'864') IMS 信息头结构无效。

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') 消息组未完成。

MQRC_INCOMPLETE_MSG
(2242, X'8C2') 逻辑消息未完成。

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889') 持久性规范不一致。

MQRC_INCONSISTENT_UOW
(2245, X'8C5') 工作单元规范不一致。

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') 全局工作单元与本地工作单元冲突。

MQRC_MD_ERROR
(2026, X'7EA') 消息描述符无效。

MQRC_MDE_ERROR
(2248, X'8C8') 消息描述符扩展无效。

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') 缺少应答队列或使用了 MQPMO_SUPPRESS_REPLYTO

MQRC_MISSING_WIH
(2332, X'91C') 消息数据未以 MQWIH 开头。

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') 消息标志无效。

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') 消息序号无效。

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') 消息长度大于队列的最大长度。

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') 消息长度大于队列管理器的最大长度。

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') 消息描述符中的消息类型无效。

MQRC_MULTIPLE_原因
(2136, X'858') 返回了多个原因码。

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') 没有可用的目标队列。

MQRC_NOT_OPEN_FOR_OUTPUT
(2039, X'7F7') 未打开队列以进行输出。

MQRC_NOT_OPEN_FOR_PASS_ALL
(2093, X'82D') 队列未打开以传递所有上下文。

MQRC_NOT_OPEN_FOR_PASS_制
(2094, X'82E') 未打开队列以传递身份上下文。

MQRC_NOT_OPEN_FOR_SET_ALL
(2095, X'82F') 未打开队列以用于设置所有上下文。

MQRC_NOT_OPEN_FOR_SET_制
(2096, X'830 ') 未针对设置的身份上下文打开队列。

MQRC_OBJECT_CHANGED
(2041, X'7F9') 对象定义自打开以来已更改。

MQRC_OBJECT_DAMAGED
(2101, X'835 ') 对象已损坏。

MQRC_OFFSET_ERROR
(2251, X'8CB') 消息段偏移无效。

MQRC_OPEN_FAILED
(2137, X'859') 对象未成功打开。

MQRC_OPTIONS_ERROR
(2046, X'7FE') 选项无效或不一致。

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') 原始长度无效。

MQRC_PAGESET_ERROR
(2193, X'891 ') 访问页集数据集时出错。

MQRC_PAGESET_FULL
(2192, X'890 ') 外部存储介质已满。

MQRC_PCF_ERROR
(2149, X'865 ') PCF 结构无效。

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') 持久性无效。

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') 队列不支持持久消息。

MQRC_PMO_ERROR
(2173, X'87D') Put-message 选项结构无效。

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') 放置消息记录标志无效。

MQRC_PRIORITY_ERROR
(2050, X'802 ') 消息优先级无效。

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') 该出版物尚未传递给任何订户。

MQRC_PUT_禁止
(2051, X'803 ') 对队列, 此队列解析到的队列或主题禁止 Put 调用。

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') 放置消息记录无效。

MQRC_PUT_NOT_留存
(2479, X'09AF') 无法保留发布

MQRC_Q_DELETED
(2052, X'804 ') 队列已删除。

MQRC_Q_FULL
(2053, X'805 ') 队列已包含最大消息数。

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING
(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR STOPPING
(2162, X'872') 队列管理器正在关闭。

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') 磁盘上没有可用于队列的空间。

MQRC_RECONNECT_FAILED
(2548, X'9F4') 重新连接后, 恢复可重新连接的连接的句柄时发生错误。

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') 存在的记录数无效。

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') 消息描述符中的报告选项无效。

MQRC_RESOURCE_PROBLEM
(2102, X'836') 没有足够系统资源可用。

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') 响应记录无效。

MQRC_RFH_ERROR
(2334, X'91E') MQRFH 或 MQRFH2 结构无效。

MQRC_RMH_ERROR
(2220, X'8AC') 参考消息头结构无效。

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') 消息段中数据的长度为零。

MQRC_SEGMENTS_NOT_SUPPORTED
(2365, X'93D') 分段不受支持。

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') 发布的可能订户已存在, 但队列管理器无法检查是否将发布发送给该订户。

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') 集群工作负载出口拒绝调用。

MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') 存储类错误。

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') 外部存储介质已满。

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') 出口程序禁止调用。

MQRC_SYNCPOINT_LIMIT_已达到
(2024, X'7E8') 无法在当前工作单元中处理更多消息。

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') 同步点支持不可用。

MQRC_TM_ERROR
(2265, X'8D9') 触发器消息结构无效。

MQRC_TMC_ERROR
(2191, X'88F') 字符触发器消息结构无效。

MQRC_UNEXPECTED_ERROR
(2195, X'893') 发生了意外错误。

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932 ') 在全局工作单元中登记失败。

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933 ') 不支持混合工作单元调用。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 工作单元不可供队列管理器使用。

MQRC_WIH_ERROR

(2333, X'91D') MQWIH 结构无效。

MQRC_不法_md_version

(2257, X'8D1') 提供的 MQMD 版本不正确。

MQRC_XQH_ERROR

(2260, X'8D4') 传输队列头结构无效。

有关这些代码的详细信息，请参阅[原因码](#)。

主题使用说明

1. 以下说明适用于主题的使用:

- a. 当使用 MQPUT 来发布主题上的消息时，如果该主题的一个或多个订户由于其订户队列存在问题 (例如已满) 而无法获得发布，那么返回到 MQPUT 调用的原因码和传递行为取决于 TOPIC 上的 PMSGDLV 或 NPMSGDLV 属性的设置。请注意，当指定了 MQRO_DEAD_LETTER_Q 时，将发布内容传递到死信队列，或者当指定了 MQRO_DISCARD_MSG 时，将废弃消息视为成功传递消息。如果未交付任何发布，那么 MQPUT 将返回 MQRC_PUBLICATION_FAILURE。在下列情况下就会发生上述情况:

- 将消息发布到将 PMSGDLV 或 NPMSGDLV (取决于消息的持久性) 设置为 ALL 的 TOPIC，并且任何预订 (持久或非持久) 都具有无法接收发布的队列。
- 将消息发布到将 PMSGDLV 或 NPMSGDLV (取决于消息的持久性) 设置为 ALLDUR 的 TOPIC，并且持久预订具有无法接收发布的队列。

MQPUT 可以返回 MQRC_NONE，即使在以下情况下无法将发布交付到某些订户也是如此:

- 将消息发布到将 PMSGDLV 或 NPMSGDLV (取决于消息的持久性) 设置为 ALLAVAIL 的 TOPIC，并且任何持久或非持久预订都具有无法接收发布的队列。
- 将消息发布到将 PMSGDLV 或 NPMSGDLV (取决于消息的持久性) 设置为 ALLDUR 的 TOPIC，并且非持久预订具有无法接收发布的队列。

您可以使用 USED LQ 主题属性来确定当发布消息无法传递到其正确的订户队列时是否使用死信队列。有关使用 USED LQ 的更多信息，请参阅 [DEFINE TOPIC](#)。

- b. 如果没有正在使用的主题的订户，那么不会将已发布的消息发送到任何队列并将其废弃。不管消息是持久的还是非持久的，还是具有无限的到期时间，如果没有订户，那么仍会将其废弃。此情况的例外情况是要保留消息，在这种情况下，虽然不会将消息发送到任何订户的队列，但会针对要传递到任何新预订或要求使用 MQSUBRQ 进行保留发布的任何订户的主题进行存储。

MQPUT 和 MQPUT1

您可以使用 MQPUT 和 MQPUT1 调用将消息放入队列; 要使用的调用取决于环境

- 使用 MQPUT 调用将多条消息放在同一队列上。

首先发出指定 MQOO_OUTPUT 选项的 MQOPEN 调用，然后发出一个或多个 MQPUT 请求以向队列添加消息; 最后使用 MQCLOSE 调用关闭队列。这将提供比重复使用 MQPUT1 调用更好的性能。

- 使用 MQPUT1 调用仅将一条消息放入队列中。

此调用将 MQOPEN，MQPUT 和 MQCLOSE 调用封装到单个调用中，从而最大限度减少必须发出的调用数。

目标队列

以下说明适用于目标队列的使用:

1. 如果应用程序在不使用消息组的情况下将消息序列放在同一队列上，如果满足详细条件，那么将保留这些消息的顺序。某些条件同时适用于本地和远程目标队列; 其他条件仅适用于远程目标队列。

适用于本地和远程目标队列的条件

- 所有 MQPUT 调用都在同一工作单元中，或者都不在工作单元中。

请注意，在将消息放入单个工作单元中的特定队列时，来自其他应用程序的消息可能与队列上的消息序列相互交织。

- 所有 MQPUT 调用都是使用同一对象句柄 *Hobj* 进行的。

在某些环境中，如果调用来自同一应用程序，那么在使用不同的对象句柄时，也会保留消息序列。同一应用程序的含义由环境确定：

- 在 z/OS 上，应用程序为：
 - 对于 CICS，CICS 任务
 - 对于 IMS，此任务
 - 对于 z/OS 批处理，此任务
- 在 IBM i 上，应用程序是作业。
- 在 Windows 和 UNIX 系统上，应用程序是线程。
- 所有消息都具有相同的优先级。
- 不会将消息放入指定了 MQOO_BIND_NOT_FIXED 的集群队列中 (或者当 DefBind 队列属性的值为 MQBND_BIND_NOT_FIXED 时，MQOO_BIND_AS_Q_DEF 生效)。

适用于远程目标队列的其他条件

- 从发送队列管理器到目标队列管理器只有一条路径。

如果序列中的某些消息可能采用不同的路径 (例如，由于重新配置，流量均衡或基于消息大小的路径选择)，那么无法保证消息在目标队列管理器中的顺序。

- 消息不会临时放置在发送队列管理器，中间队列管理器或目标队列管理器上的死信队列上。

如果一个或多个消息临时放在死信队列上 (例如，由于传输队列或目标队列暂时已满)，那么这些消息可以按顺序到达目标队列。

- 这些消息都是持久消息或所有非持久消息。

如果发送队列管理器与目标队列管理器之间的路由上的通道将其 *NonPersistentMsgSpeed* 属性设置为 MQNPMS_FAST，那么非持久消息可能会跳转至持久消息之前，从而导致持久消息相对于未保留非持久消息的顺序。但是，将保留相对于彼此的持久消息和相对于彼此的非持久消息的顺序。

如果不满足这些条件，您可以使用消息组来保留消息顺序，但这需要发送和接收应用程序都使用消息分组支持。有关消息组的更多信息，请参阅：

- [MQMD- MsgFlags 字段](#)
- [MQPMO_LOGICAL_ORDER](#)
- [MQGMO_LOGICAL_ORDER](#)

分发列表

以下说明适用于分发列表的使用。

在以下环境中支持分发列表: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机。

1. 您可以使用 version-1 或 version-2 MQPMO 将消息放入分发列表。如果使用 version-1 MQPMO (或 *RecsPresent* 等于零的 version-2 MQPMO)，那么应用程序无法提供放置消息记录或响应记录。如果成功将消息发送到分发列表中的某些队列而不是其他队列，那么无法识别迂到错误的队列。

如果应用程序提供了放置消息记录或响应记录，请将 *Version* 字段设置为 MQPMO_VERSION_2。

您还可以使用 version-2 MQPMO 通过确保 *RecsPresent* 为零，将消息发送到不在分发列表中的单个队列。

2. 完成代码和原因码参数设置如下：

- 如果对分发列表中的队列的放入都以相同方式成功或失败，那么将设置完成代码和原因码参数以描述公共结果。在这种情况下，未设置 MQRR 响应记录 (如果由应用程序提供)。

例如，如果每个 put 成功，那么完成代码和原因码将设置为 MQCC_OK 和 MQRC_NONE；如果每个 put 失败，因为所有队列都禁止 put，那么参数将设置为 MQCC_FAILED 和 MQRC_PUT_禁止。

- 如果对分发列表中的队列的放置并非全部成功或以相同方式失败：
 - 如果至少有一个 put 成功，那么完成代码参数设置为 MQCC_WARNING，如果所有 put 失败，那么设置为 MQCC_FAILED。
 - 原因码参数设置为 MQRC_MULTIPLE_REASON。
 - 响应记录 (如果由应用程序提供) 将设置为分发列表中队列的各个完成代码和原因码。

如果由于该目标的打开失败而导致放入目标失败，那么响应记录中的字段将设置为 MQCC_FAILED 和 MQRC_OPEN_FAILED；该目标包含在 *InvalidDestCount* 中。

3. 如果分发列表中的目标解析为本地队列，那么消息将以正常格式 (即，不作为分发列表消息) 放置在该队列上。如果多个目标解析为同一本地队列，那么将在队列中针对每个此类目标放置一条消息。

如果分发列表中的目标解析为远程队列，那么会将消息放置在相应的传输队列上。如果多个目标解析为同一个传输队列，那么可以将包含这些目标的单个分发列表消息放在传输队列上，即使这些目标在应用程序提供的目标列表中不相邻也是如此。但是，仅当传输队列支持分发列表消息时，才能执行此操作 (请参阅 *DistLists*)。

如果传输队列不支持分发列表，那么将在使用该传输队列的每个目标的传输队列上放置一个正常格式的消息副本。

如果具有应用程序消息数据的分发列表对于传输队列过大，那么分发列表消息将拆分为更小的分发列表消息，每个消息包含更少的目标。如果应用程序消息数据仅适合队列，那么根本无法使用分发列表消息，并且队列管理器会为使用该传输队列的每个目标生成一个正常格式的消息副本。

如果不同目标具有不同的消息优先级或消息持久性 (当应用程序指定 MQPRI_PRIORITY_AS_Q_DEF 或 MQPER_PERSISTENCE_AS_Q_DEF 时，可能会发生此情况)，那么消息不会保存在同一分发列表消息中。相反，队列管理器会根据需要生成尽可能多的分发列表消息，以适应不同的优先级和持久性值。

4. 放入分发列表可能导致：

- 单个分发列表消息，或者
- 一些较小的分发列表消息，或者
- 将分发列表消息与正常消息混合使用，或者
- 仅正常消息。

发生上述情况的原因取决于是否：

- 列表中的目标是本地的，远程的或混合的。
- 目标具有相同的消息优先级和消息持久性。
- 传输队列可以保存分发列表消息。
- 传输队列的最大消息长度足以容纳分发列表形式的消息。

但是，无论发生上述哪种情况，在以下情况下，生成的每条物理消息 (即，由 put 生成的每条正常消息或分发列表消息) 都仅算作一条消息：

- 检查应用程序是否已超过工作单元中允许的最大消息数 (请参阅 *MaxUncommittedMsgs* 队列管理器属性)。
- 正在检查是否满足触发条件。
- 增大队列深度并检查是否将超过队列的最大队列深度。

5. 如果对队列定义的任何更改 (例如，解析路径中的更改) 会导致在单独打开队列时句柄变为无效，那么这些更改不会导致分发列表句柄变为无效。但是，在后续 MQPUT 调用上使用分发列表句柄时，会导致该特定队列发生故障。

头

如果在应用程序消息数据开始时将消息与一个或多个 WebSphere MQ 头结构放在一起，那么队列管理器将对头结构执行某些检查以验证它们是否有效。如果队列管理器检测到错误，那么调用将失败并产生相应的原因码。所执行的检查根据存在的特定结构而有所不同：

- 仅当在 MQPUT 或 MQPUT1 调用上使用 version-2 或更高版本的 MQMD 时，才会执行检查。如果使用了 version-1 MQMD，那么不会执行检查，即使消息数据开头存在 MQMDE 也是如此。
- 不会验证本地队列管理器不支持的结构以及消息中第一个 MQDLH 之后的结构。
- MQDH 和 MQMDE 结构由队列管理器完全验证。
- 其他结构由队列管理器部分验证 (并非检查所有字段)。

队列管理器执行的常规检查包括以下内容：

- *StrucId* 字段必须有效。
- *Version* 字段必须有效。
- *StrucLength* 字段必须指定足够大的值，以包含结构以及构成结构一部分的任何变长数据。
- *CodedCharSetId* 字段不得为零，也不得为无效的负值 (在大多数 WebSphere MQ 头结构中，MQCCSI_DEFAULT，MQCCSI_嵌入式，MQCCSI_Q_MGR 和 MQCCSI_UNDEFINED 都无效)。
- 调用的 *BufferLength* 参数必须指定足以包含该结构的值 (该结构不得超出消息末尾)。

除了对结构进行常规检查外，还必须满足以下条件：

- PCF 消息中结构的长度总和必须等于 MQPUT 或 MQPUT1 调用上的 *BufferLength* 参数指定的长度。PCF 消息是格式名为 MQFMT_ADMIN，MQFMT_EVENT 或 MQFMT_PCF 的消息。
- 不得截断 WebSphere MQ 结构，除非在允许截断结构的以下情况下：
 - 报告消息的消息。
 - PCF 消息。
 - 包含 MQDLH 结构的消息。(可以截断第一个 MQDLH 之后的结构; 不能截断 MQDLH 之前的结构。)
- WebSphere MQ 结构不得拆分为两个或多个段; 该结构必须完全包含在一个段中。

缓冲区

对于 Visual Basic 编程语言，以下要点适用：

- 如果 *Buffer* 参数的大小小于 *BufferLength* 参数指定的长度，那么调用将失败，原因码为 MQRC_BUFFER_LENGTH_ERROR。
- *Buffer* 参数声明为 String 类型。如果要放置在队列上的数据的类型不是 String，请使用 MQPUTAny 调用代替 MQPUT。

MQPUTAny 调用与 MQPUT 调用具有相同的参数，但 *Buffer* 参数声明为 Any 类型，允许将任何类型的数据放入队列中。但是，这意味着无法检查 *Buffer* 以确保其大小至少为 *BufferLength* 字节。

C 调用

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,  
&CompCode, &Reason);
```

按如下所示声明参数：

```
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 */
```

COBOL 调用

```
CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,  
BUFFER, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.
```

PL/I 调用

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,  
CompCode, Reason);
```

按如下所示声明参数:

```
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 */
```

高级汇编程序调用

```
CALL MQPUT, (HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH, X  
BUFFER, COMPCODE, REASON)
```

按如下所示声明参数:

| | | | |
|--------------|---------|-------|--|
| 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 |

Visual Basic 调用

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode, Reason
```

按如下所示声明参数:

```
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 -放置一条消息

MQPUT1 调用将一条消息放入队列或分发列表中, 或放入主题中。

队列, 分发列表或主题不需要打开。

语法

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNEX 调用返回了 *Hconn* 的值。

在 CICS 应用程序的 z/OS 上, 以及在以兼容性方式运行的应用程序的 IBM i 上, 可以省略 MQCONN 调用, 并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

ObjDesc

类型:MQOD-输入/输出

这是一种结构, 用于标识将消息添加到的队列或将消息发布到的主题。请参阅第 414 页的『MQOD-对象描述符』, 以了解详细信息。

如果结构是队列, 那么必须授权用户打开队列以进行输出。该队列 **不能** 是模型队列。

MsgDesc

类型:MQMD-输入/输出

此结构描述要发送的消息的属性, 并在 put 请求完成后接收反馈信息。请参阅第 364 页的『MQMD - 消息描述符』, 以了解详细信息。

如果应用程序提供了 version-1 MQMD, 那么可以使用 MQMDE 结构作为消息数据的前缀, 以指定 version-2 MQMD 中存在但 version-1 中不存在的字段的值。将 MQMD 中的 格式 字段设置为 MQFMT_MD_EXTENSION, 以指示存在 MQMDE。请参阅第 406 页的『MQMDE-消息描述符扩展』以获取更多详细信息。

如果在 MQGMO 结构的 *MsgHandle* 字段中提供了有效消息句柄, 或者在 MQPMO 结构的 *OriginalMsgHandle* 或 *NewMsgHandle* 字段中提供了有效消息句柄, 那么应用程序不需要提供 MQMD 结构。如果在其中一个字段中未提供任何内容, 那么将从与消息句柄关联的描述符中获取消息描述符。

PutMsg 选项

类型:MQPMO-输入/输出

请参阅第 433 页的『MQPMO-Put-message 选项』，以了解详细信息。

BufferLength

类型: MQLONG-输入

Buffer 中消息的长度。零有效，指示消息不包含应用程序数据。上限取决于各种因素; 有关更多详细信息，请参阅 MQPUT 调用的 *BufferLength* 参数的描述。

buffer

类型: MQBYTEExBuffer 长度-输入

这是一个缓冲区，其中包含要发送的应用程序消息数据。将缓冲区与消息中数据的性质相应的边界对齐。4 字节对齐适用于大多数消息 (包括包含 WebSphere MQ 头结构的消息)，但某些消息可能需要更严格的对齐。例如，包含 64 位二进制整数的消息可能需要 8 字节对齐。

如果 *Buffer* 包含字符或数字数据，请将 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段设置为适合于数据的值; 这使消息接收方能够将数据 (如果需要) 转换为接收方使用的字符集和编码。

注: MQPUT1 调用上的所有其他参数必须采用本地队列管理器的字符集和编码 (由 *CodedCharSetId* queue-manager 属性和 MQENC_NATIVE 提供)。

在 C 编程语言中，该参数被声明为指向 void 的指针; 任何类型的数据的地址都可以被指定为该参数。

如果 *BufferLength* 参数为零，那么不会引用 *Buffer*; 在此情况下，以 C 或 System/390 汇编程序编写的程序传递的参数地址可以为空。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_MULTIPLE_原因

(2136, X'858') 返回了多个原因码。

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') 消息组未完成。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 逻辑消息未完成。

MQRC_PRIORITY_EXCEEDS_MAXIMUM

(2049, X'801') 消息优先级超过支持的最大值。

MQRC_UNKNOWN_REPORT_OPTION

(2104, X'838') 无法识别消息描述符中的报告选项。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') 无法装入适配器服务模块。

MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') 别名基本队列不是有效类型。

MQRC_API_EXIT_ERROR
(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH
(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BACKED_OUT
(2003, X'7D3') 工作单元已回退。

MQRC_BUFFER_ERROR
(2004, X'7D4') 缓冲区参数无效。

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') 缓冲区长度参数无效。

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CF_NOT_AVAILABLE
(2345, X'929') 耦合设施不可用。

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') 耦合设施结构授权检查失败。

MQRC_CF_STRUC_ERROR
(2349, X'92D') 耦合设施结构无效。

MQRC_CF_STRUC_FAILED
(2373, X'945') 耦合设施结构失败。

MQRC_CF_STRUC_IN_USE
(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') 正在使用耦合设施结构 list-header。

MQRC_CFGR_ERROR
(2416, X'970') 消息数据中的 PCF 组参数结构 MQCFGR 无效。

MQRC_CFH_ERROR
(2235, X'8BB') PCF 头结构无效。

MQRC_CFIF_ERROR
(2414, X'96E') 消息数据中的 PCF 整数过滤器参数结构无效。

MQRC_CFIL_ERROR
(2236, X'8BC') PCF 整数列表参数结构或 PCIF*64 整数列表参数结构无效。

MQRC_CFIN_ERROR
(2237, X'8BD') PCF 整数参数结构或 PCIF*64 整数参数结构无效。

MQRC_CFSF_ERROR
(2415, X'96F') 消息数据中的 PCF 字符串过滤器参数结构无效。

MQRC_CFSL_ERROR
(2238, X'8BE') PCF 字符串列表参数结构无效。

MQRC_CFST_ERROR
(2239, X'8BF') PCF 字符串参数结构无效。

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') 集群工作负载出口失败。

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') 集群名称解析失败。

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') 集群资源错误。

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') COD 报告选项对于 XCF 队列无效。

MQRC_CONNECTION_BROKEN
(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') 未授权连接。

MQRC_CONNECTION_QUIESCING
(2202, X'89A') 连接正在停顿。

MQRC_CONNECTION_STOPPING
(2203, X'89B') 连接正在关闭。

MQRC_CONTENT_ERROR
2554 (X'09FA) 无法解析消息内容以确定是否可以使用扩展消息选择器将消息传递到订户。

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831 ') 引用的队列句柄不保存上下文。

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') 上下文不可用于引用的队列句柄。

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') 数据长度参数无效。

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') DB2 子系统不可用。

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') 缺省传输队列不是本地传输队列。

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897 ') 缺省传输队列使用错误。

MQRC_DH_ERROR
(2135, X'857 ') 分发头结构无效。

MQRC_DLH_ERROR
(2141, X'85D') 死信头结构无效。

MQRC_EPH_ERROR
(2420, X' 974 ') 嵌入式 PCF 结构无效。

MQRC_EXPIRY_ERROR
(2013 年, X'7DD') 到期时间无效。

MQRC_FEEDBACK_ERROR
(2014, X'7DE') 反馈代码无效。

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') 全局工作单元冲突。

MQRC_GROUP_ID_ERROR
(2258, X'8D2') 组标识无效。

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X' 931 ') 用于全局工作单元的句柄。

MQRC_HANDLE_NOT_AVAILABLE
(2017 年, X'7E1') 没有更多可用的句柄。

MQRC_HCONN_ERROR
(2018, X'7E2') 连接句柄无效。

MQRC_HEADER_ERROR
(2142, X'85E') WebSphere MQ 头结构无效。

MQRC_IH_ERROR
(2148, X'864') IMS 信息头结构无效。

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') 全局工作单元与本地工作单元冲突。

MQRC_MD_ERROR
(2026, X'7EA') 消息描述符无效。

MQRC_MDE_ERROR
(2248, X'8C8') 消息描述符扩展无效。

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') 缺少应答队列。

MQRC_MISSING_WIH
(2332, X'91C') 消息数据未以 MQWIH 开头。

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') 消息标志无效。

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') 消息序号无效。

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') 消息长度大于队列的最大长度。

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') 消息长度大于队列管理器的最大长度。

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') 消息描述符中的消息类型无效。

MQRC_MULTIPLE_原因
(2136, X'858') 返回了多个原因码。

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') 没有可用的目标队列。

MQRC_NOT_AUTHORIZED
(2035, X'7F3') 未获得访问授权。

MQRC_OBJECT_DAMAGED
(2101, X'835') 对象已损坏。

MQRC_OBJECT_IN_USE
(2042, X'7FA') 对象已打开，但有冲突的选项。

MQRC_OBJECT_LEVEL_不兼容
(2360, X'938') 对象级别不兼容。

MQRC_OBJECT_NAME_ERROR
(2152, X'868') 对象名无效。

MQRC_OBJECT_NOT_UNIQUE
(2343, X'927') 对象不唯一。

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869') 对象队列管理器名称无效。

MQRC_OBJECT_RECORDS_ERROR
(2155, X'86B') 对象记录无效。

MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') 对象类型无效。

MQRC_OD_ERROR
(2044, X'7FC') 对象描述符结构无效。

MQRC_OFFSET_ERROR
(2251, X'8CB') 消息段偏移无效。

MQRC_OPTIONS_ERROR
(2046, X'7FE') 选项无效或不一致。

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') 原始长度无效。

MQRC_PAGESET_ERROR
(2193, X'891') 访问页集数据集时出错。

MQRC_PAGESET_FULL
(2192, X'890') 外部存储介质已满。

MQRC_PCF_ERROR
(2149, X'865') PCF 结构无效。

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') 持久性无效。

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800') 队列不支持持久消息。

MQRC_PMO_ERROR
(2173, X'87D') Put-message 选项结构无效。

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') 放置消息记录标志无效。

MQRC_PRIORITY_ERROR
(2050, X'802') 消息优先级无效。

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') 该出版物尚未传递给任何订户。

MQRC_PUT_禁止
(2051, X'803') 对队列禁止 Put 调用。

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') 放置消息记录无效。

MQRC_Q_DELETED
(2052, X'804') 队列已删除。

MQRC_Q_FULL
(2053, X'805') 队列已包含最大消息数。

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR QUIESCING
(2161, X'871') 队列管理器正在停顿。

MQRC_Q_MGR_STOPPING
(2162, X'872') 队列管理器正在关闭。

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808') 磁盘上没有可用于队列的空间。

MQRC_Q_TYPE_ERROR
(2057, X'809') 队列类型无效。

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') 存在的记录数无效。

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888') 远程队列名无效。

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') 消息描述符中的报告选项无效。

MQRC_RESOURCE_PROBLEM
(2102, X'836') 没有足够系统资源可用。

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') 响应记录无效。

MQRC_RFH_ERROR
(2334, X'91E') MQRFH 或 MQRFH2 结构无效。

MQRC_RMH_ERROR
(2220, X'8AC') 参考消息头结构无效。

MQRC_SECURITY_ERROR
(2063, X'80F') 发生了安全性错误。

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') 消息段中数据的长度为零。

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') 发布的可能订户已存在, 但队列管理器无法检查是否将发布发送给该订户。

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') 集群工作负载出口拒绝调用。

MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') 存储类错误。

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') 外部存储介质已满。

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') 出口程序禁止调用。

MQRC_SYNCPOINT_LIMIT_已达到
(2024, X'7E8') 无法在当前工作单元中处理更多消息。

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') 同步点支持不可用。

MQRC_TM_ERROR
(2265, X'8D9') 触发器消息结构无效。

MQRC_TMC_ERROR
(2191, X'88F') 字符触发器消息结构无效。

MQRC_UNEXPECTED_ERROR
(2195, X'893') 发生了意外错误。

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') 未知别名基本队列。

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') 未知缺省传输队列。

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') 未知对象名。

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') 未知对象队列管理器。

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') 未知远程队列管理器。

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') 未知传输队列。

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932 ') 在全局工作单元中登记失败。

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933 ') 不支持混合工作单元调用。

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') 工作单元不可供队列管理器使用。

MQRC_WIH_ERROR
(2333, X'91D') MQWIH 结构无效。

MQRC_不法_cf_level

(2366, X'93E') 耦合设施结构级别错误。

MQRC_不法_md_version

(2257, X'8D1') 提供的 MQMD 版本不正确。

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') 传输队列不是本地的。

MQRC_XMIT_Q_USAGE_ERROR

(2092, X'82C') 传输队列使用错误。

MQRC_XQH_ERROR

(2260, X'8D4') 传输队列头结构无效。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

1. MQPUT 和 MQPUT1 调用都可用于将消息放入队列; 要使用的调用取决于以下情况:

- 使用 MQPUT 调用将多条消息放在同一队列上。

首先发出指定 MQOO_OUTPUT 选项的 MQOPEN 调用，然后发出一个或多个 MQPUT 请求以向队列添加消息; 最后使用 MQCLOSE 调用关闭队列。这将提供比重复使用 MQPUT1 调用更好的性能。

- 使用 MQPUT1 调用仅将一条消息放入队列中。

此调用将 MQOPEN, MQPUT 和 MQCLOSE 调用封装到单个调用中，从而最大限度减少必须发出的调用数。

2. 如果应用程序在不使用消息组的情况下将消息序列放在同一队列上，如果满足特定条件，那么将保留这些消息的顺序。但是，在大多数环境中，MQPUT1 调用不满足这些条件，因此不会保留消息顺序。必须改为在这些环境中使用 MQPUT 调用。请参阅 [MQPUT 使用说明](#) 以获取详细信息。

3. MQPUT1 调用可用于将消息放入分发列表。有关此操作的常规信息，请参阅 MQOPEN 和 MQPUT 调用的用法说明。

在以下环境中支持分发列表: AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ 客户机。

使用 MQPUT1 调用时，以下差异适用:

- a. 如果应用程序提供 MQRR 响应记录，那么必须使用 MQOD 结构提供这些记录; 不能使用 MQPMO 结构提供这些记录。
- b. 响应记录中的 MQPUT1 从不返回原因码 MQRC_OPEN_FAILED; 如果队列未能打开，那么该队列的响应记录包含由打开操作生成的原因码。

如果队列的打开操作成功，并且完成代码为 MQCC_WARNING，那么该队列的响应记录中的完成代码和原因码将替换为由放置操作生成的完成代码和原因码。

与 MQOPEN 和 MQPUT 调用一样，仅当调用结果对于分发列表中的所有队列都不相同时，队列管理器才会设置响应记录 (如果提供); 这由调用完成时使用原因码 MQRC_MULTIPLE_REASON 指示。

4. 如果使用 MQPUT1 调用将消息放在集群队列上，那么该调用的行为就像在 MQOPEN 调用上指定了 MQOO_BIND_NOT_FIXED 一样。

5. 如果消息在应用程序消息数据的开头放置了一个或多个 WebSphere MQ 头结构，那么队列管理器将对头结构执行某些检查以验证它们是否有效。有关此操作的更多信息，请参阅 MQPUT 调用的用法说明。

6. 如果出现多个警告情境 (请参阅 *CompCode* 参数)，那么返回的原因码是以下列表中适用的第一个原因码:

- a. MQRC_MULTIPLE_原因
- b. MQRC_INCOMPLETE_MSG
- c. MQRC_INCOMPLETE_GROUP
- d. MQRC_PRIORITY_EXCEEDS_MAXIMUM 或 MQRC_UNKNOWN_REPORT_OPTION

7. 对于 Visual Basic 编程语言，以下要点适用:

- 如果 *Buffer* 参数的大小小于 *BufferLength* 参数指定的长度，那么调用将失败，原因码为 MQRC_BUFFER_LENGTH_ERROR。
- *Buffer* 参数声明为 *String* 类型。如果要放置在队列上的数据的类型不是 *String*，请使用 MQPUT1Any 调用代替 MQPUT1。

MQPUT1Any 调用具有与 MQPUT1 调用相同的参数，但 *Buffer* 参数声明为 *Any* 类型，允许将任何类型的数据放在队列上。但是，这意味着无法检查 *Buffer* 以确保其大小至少为 *BufferLength* 字节。

8. 发出带有 MQPMO_SYNCPOINT 的 MQPUT1 调用时，缺省行为会发生更改，因此 put 操作将异步完成。这可能导致依靠返回的 MQOD 和 MQMD 结构中的某些字段的某些应用程序行为发生更改（但是现在这些字段现在包含未定义的值）。应用程序可以指定 MQPMO_SYNC_RESPONSE 以确保同步执行 put 操作并完成所有相应的字段值。

C 调用

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
        BufferLength, Buffer, &CompCode, &Reason);
```

按如下所示声明参数：

```
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 */
```

COBOL 调用

```
CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
                   BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

按如下所示声明参数：

```
** 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.
```

PL/I 调用

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
             CompCode, Reason);
```

按如下所示声明参数：

```

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 */

```

高级汇编程序调用

```

CALL MQPUT1, (HCONN, OBJDESC, MSGDESC, PUTMSGOPTS, BUFFERLENGTH, X
             BUFFER, COMPCODE, REASON)

```

按如下所示声明参数:

| | | | |
|--------------|---------|-------|---|
| 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 |

Visual Basic 调用

```

MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
       CompCode, Reason

```

按如下所示声明参数:

```

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-设置对象属性

使用 MQSET 调用来更改由句柄表示的对象的属性。对象必须是队列。

语法

```

MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength,
       CharAttrs, Compcode, Reason)

```

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNEX 调用返回了 Hconn 的值。

在 CICS 应用程序的 z/OS 上,以及在以兼容性方式运行的应用程序的 IBM i 上,可以省略 MQCONN 调用,并为 Hconn 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

Hobj

类型:MQHOBJ-输入

此句柄表示具有要设置的属性的队列对象。该句柄由先前指定了 MQOO_SET 选项的 MQOPEN 调用返回。

SelectorCount

类型:MQLONG-输入

这是 *Selectors* 数组中提供的选择器的计数。这是要设置的属性数。零是有效值。允许的最大数目为 256。

选择器

类型:MQLONGxSelector 计数-输入

这是 *SelectorCount* 属性选择器的数组; 每个选择器标识一个具有要设置的值的属性 (整数或字符)。

对于 *Hobj* 表示的队列类型, 每个选择器都必须有效。仅允许某些 MQIA_* 和 MQCA_* 值; 如后列出。

可以按任何顺序指定选择器。对应于整数属性选择器 (MQIA_* 选择器) 的属性值必须以这些选择器在 *Selectors* 中出现的相同顺序在 *IntAttrs* 中指定。与字符属性选择器 (MQCA_* 选择器) 对应的属性值必须在 *CharAttrs* 中按这些选择器出现的顺序指定。MQIA_* 选择器可以与 MQCA_* 选择器交互; 只有每种类型中的相对顺序很重要。

您可以多次指定同一选择器; 如果指定了同一选择器, 那么为特定选择器指定的最后一个值将生效。

注:

1. 在两个不同的范围内分配整数和字符属性选择器: MQIA_* 选择器位于 MQIA_FIRST 到 MQIA_LAST 的范围内, MQCA_* 选择器位于 MQCA_FIRST 到 MQCA_LAST 的范围内。
对于每个范围, 常量 MQIA_LAST_USED 和 MQCA_LAST_USED 定义队列管理器接受的最高值。
2. 如果首先出现所有 MQIA_* 选择器, 那么可以使用相同的元素编号来寻址 *Selectors* 和 *IntAttrs* 数组中的相应元素。
3. 如果 *SelectorCount* 参数为零, 那么不会引用 *Selectors*; 在这种情况下, 以 C 或 System/390 汇编程序编写的程序传递的参数地址可能为空。

下表中列出了可设置的属性。无法使用此调用设置其他属性。对于 MQCA_* 属性选择器, 在括号中提供了用于定义 *CharAttrs* 中所需的字符串长度 (以字节为单位) 的常量。

| 选择器 | 描述 | 注 |
|---------------------------|--------------------------------|---|
| MQCA_TRIGGER_DATA | 触发数据 (MQ_TRIGGER_DATA_LENGTH)。 | |
| MQIA_DIST_LISTS | 分发列表支持。 | 1 |
| MQIA_禁止获取 | 是否允许执行 get 操作。 | |
| MQIA_禁止放入 | 是否允许执行放置操作。 | |
| MQIA_TRIGGER_CONTROL | 触发器控制。 | |
| MQIA_TRIGGER_DEPTH | 触发器深度。 | |
| MQIA_TRIGGER_MSG_PRIORITY | 触发器的阈值消息优先级。 | |
| MQIA_TRIGGER_TYPE | 触发器类型。 | |

注:

1. 仅在 AIX, HP-UX, IBM i, Solaris, Linux, Windows 以及连接到这些系统的 WebSphere MQ MQI 客户机上受支持。

IntAttrCount

类型: MQLONG-输入

这是 *IntAttrs* 数组中的元素数, 并且必须至少是 *Selectors* 参数中的 MQIA_* 选择器数。如果没有任何值, 那么零是有效值。

IntAttrs

类型: MQLONGxIntAttrCount -输入

这是 *IntAttrCount* 整数属性值的数组。这些属性值的顺序必须与 *Selectors* 数组中的 MQIA_* 选择器的顺序相同。

如果 *IntAttrCount* 或 *SelectorCount* 参数为零, 那么不会引用 *IntAttrs*; 在这种情况下, 以 C 或 System/390 汇编程序编写的程序传递的参数地址可能为空。

CharAttr 长度

类型: MQLONG-输入

这是 *CharAttrs* 参数的长度 (以字节计), 并且必须至少是 *Selectors* 数组中指定的字符属性的长度总和。如果 *Selectors* 中没有 MQCA_* 选择器, 那么零是有效值。

CharAttrs

类型: MQCHARxCharAttrLength -输入

这是包含并置在一起的字符属性值的缓冲区。缓冲区的长度由 *CharAttrLength* 参数提供。

必须以与 *Selectors* 数组中的 MQCA_* 选择器相同的顺序指定字符属性。每个字符属性的长度是固定的 (请参阅 *Selectors*)。如果要为属性设置的值包含的非空白字符数少于定义的属性长度, 请将 *CharAttrs* 中的值填充为空白, 以使属性值与定义的属性长度相匹配。

如果 *CharAttrLength* 或 *SelectorCount* 参数为零, 那么不会引用 *CharAttrs*; 在这种情况下, 以 C 或 System/390 汇编程序编写的程序传递的参数地址可能为空。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CF_STRUC_FAILED

(2373, X'945') 耦合设施结构失败。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') 耦合设施结构正在使用中。

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') 正在使用耦合设施结构 list-header。

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6') 字符属性的长度无效。

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7') 字符属性字符串无效。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') CICS 拒绝了等待请求。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 未授权连接。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') DB2 子系统不可用。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_HOBJ_ERROR

(2019, X'7E3') 对象句柄无效。

MQRC_INHIBIT_VALUE_ERROR

(2020, X'7E4') 禁止获取或禁止放入队列属性的值无效。

MQRC_INT_ATTR_COUNT_ERROR

(2021, X'7E5') 整数属性计数无效。

MQRC_INT_ATTRS_ARRAY_ERROR

(2023, X'7E7') 整数属性数组无效。

MQRC_NOT_OPEN_FOR_SET

(2040, X'7F8') 未打开队列以进行设置。

MQRC_OBJECT_CHANGED

(2041, X'7F9') 对象定义自打开以来已更改。

MQRC_OBJECT_DAMAGED

(2101, X'835') 对象已损坏。

MQRC_PAGESET_ERROR

(2193, X'891') 访问页集数据集时出错。

MQRC_Q_DELETED

(2052, X'804') 队列已删除。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') 队列管理器名称无效或者未知。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') 队列管理器针对连接不可用。

MQRC_Q_MGR_STOPPING

(2162, X'872') 队列管理器正在关闭。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_SELECTOR_COUNT_ERROR

(2065, X'811') 选择器计数无效。

MQRC_SELECTOR_ERROR

(2067, X'813') 属性选择器无效。

已超过 MQRC_SELECTOR_LIMIT_AUTHORIZED

(2066, X'812') 选择器计数过大。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口程序禁止调用。

MQRC_TRIGGER_CONTROL_ERROR

(2075, X'81B') 触发器控制属性的值无效。

MQRC_TRIGGER_DEPTH_ERROR

(2076, X'81C') 触发器深度属性的值无效。

MQRC_TRIGGER_MSG_PRIORITY_ERR

(2077, X'81D') trigger-message-priority 属性的值无效。

MQRC_TRIGGER_TYPE_ERROR

(2078, X'81E') 触发器类型属性的值无效。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

1. 通过使用此调用，应用程序可以指定整数属性的数组和/或字符属性字符串的集合。如果未发生任何错误，那么将同时设置指定的所有属性。如果发生错误（例如，如果选择器无效，或者尝试将属性设置为无效值），那么调用将失败并且不会设置任何属性。
2. 可以使用 MQINQ 调用来确定属性值；请参阅第 619 页的『MQINQ-查询对象属性』以获取详细信息。
注：并非所有具有可使用 MQINQ 调用查询的值的属性都可以使用 MQSET 调用更改其值。例如，不能使用此调用设置 process-object 或 queue-manager 属性。
3. 在队列管理器重新启动时，会保留属性更改（临时动态队列的更改除外，这些更改在队列管理器重新启动后不会存在）。
4. 不能使用 MQSET 调用来更改模型队列的属性。但是，如果使用带有 MQOO_SET 选项的 MQOPEN 调用打开模型队列，那么可以使用 MQSET 调用来设置 MQOPEN 调用创建的动态本地队列的属性。
5. 如果要设置的对象是集群队列，那么必须存在集群队列的本地实例才能成功打开。

有关对象属性的更多信息，请参阅：

- [第 731 页的『队列的属性』](#)
- [第 759 页的『名称列表的属性』](#)
- [第 761 页的『进程定义的属性』](#)
- [第 701 页的『队列管理器的属性』](#)

C 调用

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, &CompCode, &Reason);
```

按如下所示声明参数：

```
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 */

```

COBOL 调用

```

CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                  INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                  CHARATTRS, COMPCODE, REASON.

```

按如下所示声明参数:

```

** 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.

```

PL/I 调用

```

call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
            IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);

```

按如下所示声明参数:

```

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 */

```

高级汇编程序调用

```

CALL MQSET, (HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT, X
            INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON)

```

按如下所示声明参数:

| | | | |
|----------------|----|-------|---------------------------------------|
| 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 |

Visual Basic 调用

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, CompCode, Reason
```

按如下所示声明参数:

```
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-设置消息属性

使用 MQSET 调用来设置或修改消息句柄的属性。

语法

```
MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength, Value,
Compcode, Reason)
```

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。

该值必须与用于创建 *Hmsg* 参数中指定的消息句柄的连接句柄相匹配。如果使用 MQHC_UNASSOCIATED_HCONN 创建了消息句柄,那么必须在设置消息句柄属性的线程上建立有效连接,否则调用将失败,原因码为 MQRC_CONNECTION_BROKEN。

Hmsg

类型:MQHMSG-输入

这是要修改的消息句柄。该值由先前的 MQCRTMH 调用返回。

SetProp 选项

类型:MQSMPO-输入

控制如何设置消息属性。

此结构允许应用程序指定用于控制消息属性设置方式的选项。此结构是 MQSETMP 调用上的输入参数。请参阅 [MQSMPO](#) 以获取更多信息。

名称

类型:MQCHARV-输入

这是要设置的属性的名称。

请参阅 [属性名](#) 和 [属性名限制](#)，以获取有关使用属性名的更多信息。

PropDesc

类型 :MQPD-输入/输出

此结构用于定义属性的属性，包括：

- 如果该属性不受支持，那么会发生什么情况
- 属性所属的消息上下文
- 在属性流动时将其复制到其中的消息

请参阅 [MQPD](#) 以获取有关此结构的更多信息。

类型

类型 :MQLONG-输入

要设置的属性的数据类型。可以为以下某项：

MQTYPE_BOOLEAN

布尔值。 *ValueLength* 必须为 4。

MQTYPE_BYTE_STRING

字节字符串。 *ValueLength* 必须为零或更大值。

MQTYPE_INT8

8 位带符号整数。 *ValueLength* 必须为 1。

MQTYPE_INT16

16 位带符号整数。 *ValueLength* 必须为 2。

MQTYPE_INT32

32 位带符号整数。 *ValueLength* 必须为 4。

MQTYPE_INT64

64 位带符号整数。 *ValueLength* 必须为 8。

MQTYPE_FLOAT32

32 位浮点数。 *ValueLength* 必须为 4。

注：使用 IBM COBOL for z/OS 的应用程序不支持此类型。

MQTYPE_FLOAT64

64 位浮点数。 *ValueLength* 必须为 8。

注：使用 IBM COBOL for z/OS 的应用程序不支持此类型。

MQTYPE_STRING

字符串。 *ValueLength* 必须为零或更大的值，或者特殊值 MQVL_NULL_TERMINATED。

MQTYPE_NULL

该属性存在，但具有空值。 *ValueLength* 必须为零。

ValueLength

类型 :MQLONG-输入

Value 参数中属性值的长度 (以字节计)。零仅对空值或字符串或字节字符串有效。零表示该属性存在，但该值不包含任何字符或字节。

如果 *Type* 参数设置了 MQTYPE_STRING，那么该值必须大于或等于 0 或以下特殊值：

MQVL_NULL_TERMINATED

该值由字符串中迁到的第一个空值定界。空值不包含在字符串中。如果未同时设置 MQTYPE_STRING，那么此值无效。

注：如果设置了 MQVL_NULL_TERMINATED，那么用于终止字符串的空字符是来自值的字符集的空字符。

Value

类型: MQBYTEValue 长度-输入

要设置的属性的值。缓冲区必须在适合于值中数据的性质的边界上对齐。

在 C 编程语言中，该参数被声明为指向 void 的指针；任何类型的数据的地址都可以被指定为该参数。

如果 *ValueLength* 为零，那么不会引用值。在这种情况下，以 C 或 System/390 汇编程序编写的程序传递的参数地址可以为空。

CompCode

类型：MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_RFH_FORMAT_ERROR

(2421, X'0975') 无法解析包含属性的 MQRFH2 文件夹。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') 适配器不可用。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') 无法装入适配器服务模块。

MQRC_ASID_MISMATCH

(2157, X'86D') 主 ASID (Primary ASID) 与主 ASID (home ASID) 不同。

MQRC_BUFFER_ERROR

(2004, X'07D4') 值参数无效。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') 值长度参数无效。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 在先前调用完成之前输入的 MQI 调用。

MQRC_HMSG_ERROR

(2460, X'099C') 消息句柄指针无效。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') 消息句柄已在使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') 选项无效或不一致。

MQRC_PD_ERROR

(2482, X'09B2') 属性描述符结构无效。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') 属性名无效。

MQRC_PROPERTY_TYPE_ERROR

(2473, X'09A9') 属性数据类型无效。

MQRC_PROP_NUMBER_FORMAT_ERROR

(2472, X'09A8') 在值数据中迁到数字格式错误。

MQRC_SMPO_ERROR

(2463, X'099F') 设置消息属性选项结构无效。

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') 属性名编码字符集标识无效。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

C 调用

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,  
ValueLength, &Value, &CompCode, &Reason);
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDSC, TYPE,  
VALUELENGTH, VALUE, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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 PROPDSC.  
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.
```


PL/I 调用

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,  
              Value, CompCode, Reason);
```

按如下所示声明参数:

```
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 */
```

高级汇编程序调用

```
CALL MQSETMP, (HCONN,HMSG,SETMSGHOPTS,NAME,PROPDSC,TYPE,VALUELENGTH,  
              VALUE,COMPCODE,REASON)
```

按如下所示声明参数:

| | | | |
|-------------|----------|-------|--|
| HCONN | DS | F | Connection handle |
| HMSG | DS | D | Message handle |
| SETMSGOPTS | CMQSMPOA | , | Options that control the action of MQSETMP |
| NAME | CMQCHRVA | , | Property name |
| PROPDSC | 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-检索状态信息

使用 MQSTAT 调用来检索状态信息。返回的状态信息的类型由调用上指定的 "类型" 值确定。

语法

MQSTAT (*Hconn*, *Type*, *Stat*, *Compcode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNEX 调用返回了 *Hconn* 的值。

在 CICS 应用程序的 z/OS 上,以及在以兼容性方式运行的应用程序的 IBM i 上,可以省略 MQCONN 调用,并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

类型

类型:MQLONG-输入

正在请求的状态信息的类型。 > 有效值为:

MQSTAT_TYPE_ASYNC_ERROR

返回有关先前异步放置操作的信息。

MQSTAT_TYPE_RECONNECTION

返回有关重新连接的信息。如果连接正在重新连接或未能重新连接，那么此信息描述导致连接开始重新连接的故障。

此值仅对客户机连接有效。对于其他类型的连接，调用失败，原因码为

MQRC_ENVIRONMENT_ERROR

MQSTAT_TYPE_RECONNECTION_ERROR

返回有关与重新连接相关的先前故障的信息。如果连接未能重新连接，那么此信息描述导致重新连接失败的故障。

此值仅对客户机连接有效。对于其他类型的连接，调用失败，原因码为

MQRC_ENVIRONMENT_ERROR。

stat

类型:MQSTS-输入/输出

状态信息结构。请参阅第 514 页的『MQSTS-状态报告结构』，以了解详细信息。

CompCode

类型: MQLONG - 输出

完成代码；此完成代码为以下其中一项：

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 是 MQCC_FAILED:

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口失败

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') 无法装入 API 出口。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 在先前调用完成前输入了 MQI 调用。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') 与队列管理器的连接丢失。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 连接正在关闭。

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') 请求的功能在当前环境中不可用。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_Q_MGR_STOPPING

(2162, X"872")-队列管理器正在停止

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_STAT_TYPE_ERROR

(2430, X'97E') MQSTAT 类型错误

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_STS_ERROR

(2426, X'97A') MQSTS 结构出错

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

1. 指定 MQSTAT_TYPE_ASYNC_ERROR 类型的 MQSTAT 调用将返回有关先前异步 MQPUT 和 MQPUT1 操作的信息。从 MQSTAT 调用返回时传递的 MQSTS 结构包含该连接的第一个记录的异步警告或错误信息。如果第一个错误或警告后面还有其他错误或警告，那么它们通常不会更改这些值。但是，如果完成代码为 MQCC_WARNING 时发生错误，那么将改为返回完成代码为 MQCC_FAILED 的后续故障。
2. 如果自建立连接以来或自上次调用 MQSTAT 以来未发生任何错误，那么将在 MQSTS 结构中返回 CompCode MQCC_OK 和 "原因" MQRC_NONE。
3. 已在连接句柄下处理的异步调用的计数通过三个计数器字段返回: PutSuccessCount, PutWarningCount 和 PutFailureCount。每次成功处理异步操作，发出警告或失败时，队列管理器都会增加这些计数器 (请注意，出于记帐目的，放入分发列表时，每个目标队列计数一次，而不是每个分发列表计数一次)。计数器的增量不超过最大值 AMQ_LONG_MAX。
4. 成功调用 MQSTAT 会导致重置任何先前的错误信息或计数。
5. MQSTAT 的行为取决于您提供的 MQSTAT Type 参数的值。
6. **MQSTAT_TYPE_ASYNC_ERROR**
 - a. 指定 MQSTAT_TYPE_ASYNC_ERROR 类型的 MQSTAT 调用将返回有关先前异步 MQPUT 和 MQPUT1 操作的信息。从 MQSTAT 调用返回时传递的 MQSTS 结构包含该连接的第一个记录的异步警告或错误信息。如果第一个错误或警告后面还有其他错误或警告，那么它们通常不会更改这些值。但是，如果完成代码为 MQCC_WARNING 时发生错误，那么将改为返回完成代码为 MQCC_FAILED 的后续故障。
 - b. 如果自建立连接以来或自上次调用 MQSTAT 以来未发生任何错误，那么将在 MQSTS 结构中返回 CompCode MQCC_OK 和 "原因" MQRC_NONE。
 - c. 已在连接句柄下处理的异步调用的计数通过三个计数器字段返回: PutSuccessCount, PutWarningCount 和 PutFailureCount。每次成功处理异步操作，发出警告或失败时，队列管理器都会增加这些计数器 (请注意，出于记帐目的，放入分发列表时，每个目标队列计数一次，而不是每个分发列表计数一次)。计数器的增量不超过最大值 AMQ_LONG_MAX。
 - d. 成功调用 MQSTAT 会导致重置任何先前的错误信息或计数。

MQSTAT_TYPE_RECONNECTION

假设您在重新连接期间在事件处理程序内调用 MQSTAT，并将 Type 设置为 MQSTAT_TYPE_RECONNECTION。请考虑以下示例。

客户机正在尝试重新连接或未能重新连接。

MQSTS 结构中的 CompCode 是 MQCC_FAILED，Reason 可以是 MQRC_CONNECTION_BROKEN 或 MQRC_Q_MGR QUIESCING。ObjectType 是 MQOT_Q_MGR，ObjectName 是队列管理器的名称，ObjectQMgrName 是空白。

客户机已成功完成重新连接或从未断开连接。

MQSTS 结构中的 CompCode 为 MQCC_OK，Reason 为 MQRC_NONE

后续调用 MQSTAT 将返回相同的结果。

MQSTAT_TYPE_RECONNECTION_ERROR

假设您在将 Type 设置为 MQSTAT_TYPE_RECONNECTION_ERROR 的情况下调用 MQSTAT，以响应接收到对 MQI 调用的 MQRC_RECONNECT_FAILED。请考虑以下示例。

在重新连接到其他队列管理器期间重新打开队列时发生授权失败。

MQSTS 结构中的 CompCode 是 MQCC_FAILED，Reason 是重新连接失败的原因，例如 MQRC_NOT_AUTHORIZED。ObjectType 是导致问题的对象类型，例如 MQOT_QUEUE，ObjectName 是队列的名称，ObjectQMgrName 是拥有该队列的队列管理器的名称。

重新连接期间发生套接字连接错误。

MQSTS 结构中的 CompCode 是 MQCC_FAILED，Reason 是重新连接失败的原因，例如 MQRC_HOST_NOT_AVAILABLE。ObjectType 是 MQOT_Q_MGR，ObjectName 是队列管理器的名称，ObjectQMgrName 是空白。

后续调用 MQSTAT 将返回相同的结果。

C 调用

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

按如下所示声明参数：

```
MQHCONN Hconn;          /* Connection Handle */
MQLONG StatType;        /* Status type */
MQSTS Stat;              /* Status information structure */
MQLONG CompCode;        /* Completion code */
MQLONG Reason;          /* Reason code qualifying CompCode */
```

COBOL 调用

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

按如下所示声明参数：

```
**      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.
```

PL/I 调用

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

按如下所示声明参数：

```
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 汇编程序调用

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

按如下所示声明参数：

```
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-注册预订

使用 MQSUB 调用来注册特定主题的应用程序预订。

语法

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *Compcode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。

在 CICS 应用程序的 z/OS 上,以及在以兼容性方式运行的应用程序的 IBM i 上,可以省略 MQCONN 调用,并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

SubDesc

类型:MQSD-输入/输出

这是一种结构,用于标识正在由应用程序注册的正在使用的对象。请参阅第 491 页的『MQSD-预订描述符』以获取更多信息。

Hobj

类型:MQHOBJ-输入/输出

此句柄表示为获取发送到此预订的消息而建立的访问权。这些消息可以存储在特定队列上,或者队列管理器可以在不使用特定队列的情况下管理其存储器。

要使用特定队列,您必须在创建预订时将其与预订相关联。您可以使用两种方式执行此操作:

- 通过使用 DEFINE SUB MQSC 命令并为该命令提供队列对象的名称。
- 通过在使用 MQSO_CREATE 调用 MQSUB 时提供此句柄

如果此句柄作为调用上的输入参数提供,那么它必须是使用以下至少一个选项从队列的先前 MQOPEN 调用返回的有效对象句柄:

- MQOO_INPUT_*
- MQOO_BROWSE
- MQOO_OUTPUT (如果队列是远程队列)

如果情况并非如此,那么调用将失败并返回 MQRC_HOBJ_ERROR。它不能是解析为主题对象的别名队列的对象句柄。如果是这样,那么调用将失败并返回 MQRC_HOBJ_ERROR。

如果队列管理器要管理发送到此预订的消息的存储,那么应在使用 MQSO_MANAGED 选项创建预订时进行设置。然后,队列管理器将此句柄作为调用的输出参数返回。返回的句柄称为受管句柄。如果指定了 MQHO_NONE,但未指定 MQSO_MANAGED,那么调用将失败并返回 MQRC_HOBJ_ERROR。

当队列管理器向您返回受管句柄时,可以在带有或不带浏览选项的 MQGET 或 MQCB 调用, MQINQ 调用或 MQCLOSE 上使用该句柄。不能在 MQPUT, MQSUB 和 MQSET 上使用此参数;尝试执行此操作将失败,返回 MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR 或 MQRC_NOT_OPEN_FOR_SET。

如果正在使用 MQSD 结构中的 MQSO_RESUME 选项恢复此预订,那么可以通过将 MQSO_MANAGED 设置为 MQHO_NONE 来将句柄返回到此参数中的应用程序。无论预订是否正在使用受管句柄,都可以执行此操作,并且将使用 DEFINE SUB 创建的预订与该句柄一起提供给该命令上定义的预订队列很有用。在恢复以管理方式创建的预订的情况下,将使用 MQOO_INPUT_AS_Q_DEF 和 MQOO_BROWSE 打开队列。如果需要指定其他选项,那么应用程序必须显式打开预订队列并在调用上提供对象句柄。如果打开

队列时发生问题，那么调用将失败并返回 MQRC_INVALID_DESTINATION。如果提供了 *Hobj*，那么它必须等同于原始 MQSUB 调用中的 *Hobj*。这意味着如果提供了从 MQOPEN 调用返回的对象句柄，那么该句柄必须与先前使用的队列相同。如果它不是同一队列，那么调用将失败并返回 MQRC_HOBJ_ERROR。

如果正在使用 MQSD 结构中的 MQSO_ALTER 选项更改此预订，那么可以提供其他 *Hobj*。任何已传递到队列且先前通过此参数识别的发布都将保留在该队列中，如果 *Hobj* 参数现在表示另一个队列，那么应用程序将负责检索这些消息。

该表总结了将此参数与各种预订选项配合使用的情况:

| 选项 | <i>Hobj</i> | 描述 |
|----------------------------|-------------|---|
| MQSO_CREATE + MQSO_MANAGED | 在输入时忽略 | 创建具有队列管理器管理的消息存储的预订 |
| MQSO_CREATE | 有效的对象句柄 | 创建提供特定队列作为消息目标的预订。 |
| MQSO_RESUME | MQHO_NONE | 恢复先前创建的预订 (无论其是否受管)，并使队列管理器返回对象句柄以供应用程序使用。 |
| MQSO_RESUME | 有效的匹配对象句柄 | 恢复先前创建的预订，该预订使用特定队列作为消息的目标，并使用具有特定打开选项的对象句柄。 |
| MQSO_ALTER + MQSO_MANAGED | MQHO_NONE | 更改先前使用特定队列的现有预订，因此它现在是受管预订。无法更改目标的类 (受管或不受管)。 |
| MQSO_ALTER | 有效的对象句柄 | 更改现有预订 (无论它是否受管)，以便它现在使用特定队列。未使用 MQSO_MANAGED 选项时，可以更改提供的队列，但无法更改目标的类 (受管或非受管)。 |

无论提供还是返回，都必须在后续 MQGET 或 MQCB 调用上指定 *Hobj*，这些调用要接收发送到此预订的发布消息。

当对其发出 MQCLOSE 调用时，或者当定义句柄作用域的处理单元终止时 (直到应用程序断开连接)，*Hobj* 句柄不再有效。返回的对象句柄的作用域与调用上指定的连接句柄的作用域相同。有关句柄作用域的信息，请参阅 [Hconn \(MQHCONN\)-output](#)。*Hobj* 句柄的 MQCLOSE 不会影响 *Hsub* 句柄。

Hsub

类型:MQHOBJ-输出

此句柄表示已进行的预订。它可用于两个进一步的操作:

- 可以在后续 MQSUBRQ 调用上使用此命令，以在进行预订时使用 MQSO_publicATIONS_ON_REQUEST 选项时请求发送发布。
- 可以在后续 MQCLOSE 调用上使用此参数来除去已进行的预订。发出 MQCLOSE 调用时，或者定义句柄作用域的处理单元终止时，*Hsub* 句柄将不再有效。返回的对象句柄的作用域与调用上指定的连接句柄的作用域相同。*Hsub* 句柄的 MQCLOSE 不会影响 *Hobj* 句柄。

无法将此句柄传递到 MQGET 或 MQCB 调用。必须使用 *Hobj* 参数。不能在除 MQCLOSE 或 MQSUBRQ 以外的任何 WebSphere MQ 调用上使用此句柄。将此句柄传递到任何其他 WebSphere MQ 调用会导致 MQRC_HOBJ_ERROR。

CompCode

类型: MQLONG - 输出

完成代码; 此完成代码为以下其中一项:

MQCC_OK

成功完成

MQCC_WARNING

警告 (部分完成)

MQCC_FAILED

通话失败

原因

类型: MQLONG - 输出

限定 *CompCode* 的原因码。如果 *CompCode* 为 MQCC_OK, 那么原因码如下所示:**MQRC_NONE**

(0, X'000') 没有要报告的原因。

如果 *CompCode* 是 MQCC_FAILED, 那么原因码为下列其中一项:**MQRC_CLUSTER_RESOLUTION_ERROR**

(2189, X'88D') 集群名称解析失败。

MQRC_DURABILITY_NOT_ALLOWED

2436 (X'0984 ') 使用 MQSO_持久选项的 MQSUB 调用失败。

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') 请求的功能在当前环境中不可用。

MQRC_HOBJ_ERROR

2019 (X'07E3') 对象句柄 Hobj 无效。

MQRC_IDENTITY_MATCH

2434 (X'0982 ') 预订名称与现有预订匹配。

MQRC_NOT_AUTHORIZED

2035 (X'07F3') 用户无权执行该操作。

MQRC_OBJECT_STRING_ERROR

2441 (X'0989 ') Objectstring 字段无效。

MQRC_OPTIONS_ERROR

2046 (X'07FE') "选项" 参数或字段包含无效的选项或无效的选项组合。

MQRC_Q_MGR QUIESCING

2161 (X'0871 ') 队列管理器正在停顿。

MQRC_RECONNECT_Q_MGR_REQD

2555 (X'09FB' X) 需要 MQCNO_RECONNECT_Q_MGR 选项。

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') 无法检索预订主题字符串存在的保留发布。

MQRC_RETAINED_NOT_交付

2526 (X'09DE') 对于预订主题字符串存在的保留发布, 无法传递到预订目标队列, 也无法传递到死信队列。

MQRC_SD_ERROR

2424 (X'0978 ') 预订描述符 (MQSD) 无效。

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') 选择字符串未遵循 WebSphere MQ 选择器语法, 并且没有可用的扩展消息选择提供程序。

MQRC_SELECTION_STRING_ERROR

2519 (X'09D7') 必须指定选择字符串, 如 MQCHARV 结构文档中所述。

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') 发出了 MQOPEN, MQPUT1 或 MQSUB 调用, 但指定了包含语法错误的选择字符串。

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') SubUser 数据字段无效。

MQRC_SUB_NAME_ERROR

2440 (X'0988 ') SubName 字段无效。

MQRC_SUB_ALREADY_EXISTS

2432 (X'0980 ') 预订已存在。

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') SubUser 数据字段无效。

MQRC_TOPIC_STRING_ERROR

2425 (X'0979 ') 主题字符串无效。

MQRC_UNKNOWN_OBJECT_NAME

2085 (X'0825 ') 找不到标识的对象。

有关这些代码的详细信息，请参阅[原因码](#)。

使用说明

1. 对主题进行预订，使用预定义主题对象的短名称，主题字符串的全名或由两个部分并置而成。请参阅第 491 页的『MQSD-预订描述符』中 *ObjectName* 和 *ObjectString* 的描述。
2. 在发出 MQSUB 调用时，队列管理器将执行安全性检查，以验证在允许访问之前，运行应用程序的用户标识是否具有相应的权限级别。相应的主题对象位于主题层次结构中，并对此主题对象进行权限检查以确保设置了预订权限。如果未使用 MQSO_MANAGED 选项，那么将在目标队列上进行权限检查，以确保设置输出权限。如果使用 MQSO_MANAGED 选项，那么不会对受管队列进行权限检查以进行输出或查询访问。
3. 如果未提供 Hobj 作为输入，那么 MQSUB 调用会分配两个句柄，一个对象句柄 (Hobj) 和一个预订句柄 (Hsub)。
4. 使用 MQSO_MANAGED 选项时，可以查询在 MQSUB 调用上返回的 Hobj，以找出回退阈值和过多回退重排队列名称之类的属性。您还可以查询受管队列的名称，但不得尝试直接打开此队列。
5. 可以对预订进行分组，仅允许将单个发布内容传递到预订组，即使有多个组与该发布内容匹配也是如此。使用 MQSO_GROUP_SUB 选项对预订进行分组，为了对预订进行分组，必须
 - 在同一队列管理器上使用同一指定队列 (未使用 MQSO_MANAGED 选项)-由 MQSUB 调用上的 Hobj 参数表示
 - 共享相同的 SubCorrel 标识
 - 属于同一 SubLevel

这些属性定义被视为在组中的预订集，并且也是在对预订进行分组时无法更改的属性。更改 SubLevel 会导致 MQRC_SUBLEVEL_NOT_ALTERABLE，而更改任何其他 (如果预订未分组，那么可以更改) 会导致 MQRC_GROUPING_NOT_ALTERABLE。

6. MQSD 中的字段是从使用 MQSO_RESUME 选项的 MQSUB 调用返回时填写的。返回的 MQSD 可以直接传递到 MQSUB 调用中，该调用使用 MQSO_ALTER 选项以及您需要对应用于 MQSD 的预订进行的任何更改。如表中所述，某些字段具有特殊注意事项。

| MQSUB 的 MQSD 输出 | |
|-----------------------|--|
| MQSD 中的字段名 | 特殊注意事项 |
| 访问或创建选项 | 某些选项可以在从 MQSUB 调用返回时重置。如果然后在 MQSUB 调用中复用 MQSD，那么必须显式设置所需的选项。 |
| 持久性选项，目标选项，注册选项和通配符选项 | 根据需要设置这些选项 |
| 发布选项 | 除仅适用于 MQSO_CREATE 的 MQSO_NEW_PUBLIC ICATIONS_ONLY 外，将根据需要设置这些选项。 |

| MQSUB 的 MQSD 输出 (继续) | |
|---|--|
| MQSD 中的字段名 | 特殊注意事项 |
| 其他选项 | 这些选项在从 MQSUB 调用返回时保持不变。它们控制如何发出 API 调用并且不随预订一起存储。必须在复用 MQSD 的任何后续 MQSUB 调用上根据需要进行设置。 |
| ObjectName | 此仅输入字段在从 MQSUB 调用返回时保持不变。 |
| ObjectString | 此仅输入字段在从 MQSUB 调用返回时保持不变。如果提供了缓冲区，那么将在 <i>ResObjectString</i> 字段中返回所使用的完整主题名称。 |
| AlternateUser 标识和 AlternateSecurity 标识 | 这些仅输入字段在从 MQSUB 调用返回时保持不变。它们控制如何发出 API 调用并且不随预订一起存储。必须在复用 MQSD 的任何后续 MQSUB 调用上根据需要进行设置。 |
| SubExpiry | 使用 MQSO_RESUME 选项从 MQSUB 调用返回时，此字段将设置为预订的原始到期时间，而不是剩余的到期时间。如果然后使用 MQSO_ALTER 选项在 MQSUB 调用中复用 MQSD，请重置预订到期时间以重新开始计数。 |
| SubName | 此字段是 MQSUB 调用上的输入字段，不会在输出时更改。 |
| SubUser 数据和 SelectionString | <p>如果提供了缓冲区，那么将在使用 MQSO_RESUME 选项的 MQSUB 调用的输出中返回这些可变长度字段，并在 <i>VSubSize</i> 中返回正缓冲区长度。如果未提供缓冲区，那么仅在 MQCHARV 的 <i>VSLength</i> 字段中返回长度。如果提供的缓冲区小于返回字段所需的长度，那么在提供的缓冲区中仅返回 <i>VSubSize</i> 字节。</p> <p>如果然后使用 MQSO_ALTER 选项在 MQSUB 调用中复用 MQSD，并且未提供缓冲区，但提供了非零 <i>VSLength</i>，那么如果该长度与字段的现有长度匹配，那么不会对该字段进行任何更改。</p> |
| SubCorrel 标识和 PubAccounting 令牌 | <p>如果不使用 MQSO_SET_CORREL_ID，那么队列管理器将生成 <i>SubCorrelId</i>。如果不使用 MQSO_SET_IDENTITY_CONTEXT，那么队列管理器将生成 <i>PubAccountingToken</i>。</p> <p>这些字段是在 MQSD 中使用 MQSO_RESUME 选项从 MQSUB 调用返回的。如果它们是由队列管理器生成的，那么将使用 MQSO_CREATE 或 MQSO_ALTER 选项在 MQSUB 调用上返回生成的值。</p> |
| PubPriority, SubLevel & PubApplIdentityData | 这些字段在 MQSD 中返回。 |
| ResObject 字符串 | 如果提供了缓冲区，那么仅在 MQSD 中返回此输出字段。 |

C 调用

MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)

按如下所示声明参数:

```

MQHCONN Hconn; /* Connection handle */
MQSD SubDesc; /* Subscription descriptor */
MQHOBJ Hobj; /* Object handle */
MQHOBJS Hsub; /* Subscription handle */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Reason code qualifying CompCode */

```

COBOL 调用

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

按如下所示声明参数:

```

** 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.

```

PL/I 调用

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

按如下所示声明参数:

```

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 */

```

高级汇编程序调用

```
CALL MQSUB, (HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON)
```

按如下所示声明参数:

```

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-预订请求

当订户已向 MQSO_PUBLIC ICATIONS_ON_REQUEST 注册时, 使用 MQSUBRQ 调用来发出保留发布请求。

语法

```
MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, Compcode, Reason)
```

参数

Hconn

类型 :MQHCONN-输入

此句柄表示与队列管理器的连接。先前的 MQCONN 或 MQCONNX 调用返回了 *Hconn* 的值。

在 CICS 应用程序的 z/OS 上,以及在以兼容性方式运行的应用程序的 IBM i 上,可以省略 MQCONN 调用,并为 *Hconn* 指定以下值:

MQHC_DEF_HCONN

缺省连接句柄。

Hsub

类型 :MQHOBJ-输入

此句柄表示要为其请求更新的预订。 *Hsub* 的值是从先前的 MQSUB 调用返回的。

Action

类型 :MQLONG-输入

此参数控制正在预订上请求的特定操作。必须指定以下值:

MQSR_ACTION_PUBLICATION

此操作请求针对指定主题发送更新发布。仅当订户在进行预订时在 MQSUB 调用上指定了选项 MQSO_PUBLICATIONS_ON_REQUEST 时,才能使用此选项。如果队列管理器具有主题的保留发布内容,那么会将此内容发送给订户。否则,调用将失败。如果向应用程序发送了保留的发布,那么该发布由该发布的 MQIsRetained 消息属性指示。

由于 *Hsub* 参数表示的现有预订中的主题可能包含通配符,因此订户可能会接收到多个保留发布。

SubRq 选项

类型 :MQSRO-输入/输出

这些选项控制 MQSUBRQ 的操作,请参阅第 512 页的『MQSRO-预订请求选项』以获取详细信息。

如果不需要任何选项,那么以 C 或 S/390 汇编程序编写的程序可以指定空参数地址,而不是指定 MQSRO 结构的地址。

CompCode

类型: MQLONG - 输出

完成代码;此完成代码为以下其中一项:

MQCC_OK

成功完成

MQCC_WARNING

警告(部分完成)

MQCC_FAILED

通话失败

原因

类型: MQLONG - 输出

限定 *CompCode* 的原因码。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') 请求的功能在当前环境中不可用。

MQRC_NO_RETAINED_MSG

2437 (X'0985 ') 当前没有为此主题存储的保留发布。

MQRC_OPTIONS_ERROR

2046 (X'07FE') "选项" 参数或字段包含无效的选项或无效的选项组合。

MQRC_Q_MGR QUIESCING

2161 (X'0871 ') 队列管理器正在停顿。

MQRC_SRO_ERROR

2438 (X'0986 ') 在 MQSUBRQ 调用上, "预订请求选项" MQSRO 无效。

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') 无法检索预订主题字符串存在的保留发布。

MQRC_RETAINED_NOT_交付

2526 (X'09DE') 对于预订主题字符串存在的保留发布, 无法传递到预订目标队列, 也无法传递到死信队列。

有关这些代码的详细信息, 请参阅[原因码](#)。

使用说明

以下用法说明适用于操作码 MQSR_ACTION_PUBLICATION 的使用:

1. 如果此动词成功完成, 那么与指定的预订匹配的保留发布已发送到该预订, 并且可以使用 MQGET 或 MQCB (用在创建该预订的原始 MQSUB 动词上返回的 Hobj) 来接收这些保留发布。
2. 如果创建预订的原始 MQSUB 动词所预订的主题包含通配符, 那么可以发送多个保留发布。由于此调用而发送的发布数记录在 SubRqOpts 结构的 NumPubs 字段中。
3. 如果此动词完成, 并且原因码为 MQRC_NO_RETAINED_MSG, 那么当前没有针对指定主题的保留发布。#
4. 如果此动词完成时原因码为 MQRC_RETAINED_MSG_Q_ERROR 或 MQRC_RETAINED_NOT_交付, 那么当前存在指定主题的保留发布, 但发生了错误, 这意味着无法交付这些发布。
5. 应用程序必须具有主题的当前预订, 然后才能进行此调用。如果预订是在应用程序的先前实例中进行的, 并且预订的有效句柄不可用, 那么应用程序必须首先使用 MQSO_RESUME 选项调用 MQSUB, 以获取其句柄, 以便在此调用中使用。
6. 这些发布将发送到注册为与此应用程序的当前预订配合使用的目标。如果必须将发布发送到其他位置, 那么必须首先使用带有 MQSO_ALTER 选项的 MQSUB 调用来变更预订。

C 调用

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

按如下所示声明参数:

```
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 */
```

COBOL 调用

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

按如下所示声明参数:

```
** 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.

```

PL/I 调用

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

按如下所示声明参数:

```

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 */

```

高级汇编程序调用

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

按如下所示声明参数:

```

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

```

对象的属性

此主题集合仅列出可作为 MQINQ 函数调用主题的那些 WebSphere MQ 对象, 并提供可查询的属性以及要使用的选择器的详细信息。

队列管理器的属性

某些队列管理器属性是针对特定实现固定的; 其他属性可以使用 MQSC 命令 ALTER QMGR 进行更改。

还可以使用命令 DISPLAY QMGR 来显示这些属性。可以通过打开特殊 MQOT_Q_MGR 对象并将 MQINQ 调用与返回的句柄配合使用来查询大多数队列管理器属性。

下表汇总了特定于队列管理器的属性。这些属性按字母顺序进行描述。

注: 本节中显示的属性名称是用于 MQINQ 调用的描述性名称; 这些名称与 PCF 命令的名称相同。当 MQSC 命令用于定义, 改变或显示属性时, 将使用备用短名称; 有关更多信息, 请参阅 [脚本 \(MQSC\) 命令](#)。

| 表 572: 队列管理器的属性。 | |
|--|----------------------------------|
| 具有链接和简短描述的队列管理器属性的列表 | |
| 属性 | 描述 |
| AccountingConnOverride | 覆盖记帐设置。 |
| AccountingInterval | 写入中间记帐记录的频率。 |
| ActivityConnOverride | 覆盖活动设置。 |
| ActivityTrace | 控制 WebSphere MQ MQI 应用程序活动跟踪的收集。 |
| AdoptNewMCACheck | 检查元素以确定是否采用新的 MCA。 |
| AdoptNewMCAType | 是否自动重新启动特定通道类型的 MCA 的孤立实例。 |
| AlterationDate | 上次更改定义日期 |
| AlterationTime | 上次更改定义的时间 |

表 572: 队列管理器的属性。

具有链接和简短描述的队列管理器属性的列表
(继续)

| 属性 | 描述 |
|--|---|
| AuthorityEvent | 控制是否生成授权 (未授权) 事件 |
| BridgeEvent | 网桥事件的控制属性。 |
| ChannelAutoDef | 控制是否允许自动通道定义 |
| ChannelAutoDefEvent | 控制是否生成通道自动定义事件 |
| ChannelAutoDefExit | 自动通道定义的用户出口的名称 |
| ChannelEvent | 通道事件的控制属性。 |
| ChannelInitiatorControl | 通道启动程序的控制属性 |
| ChannelMonitoring | 通道的联机监视数据 |
| ChannelStatistics | 控制通道的统计数据收集。 |
| ChinitAdapters | 用于处理 WebSphere MQ 调用的适配器子任务数。 |
| ChinitDispatchers | 要用于通道启动程序的分派器数。 |
| | 保留供 IBM 使用。 |
| ChinitTraceAutoStart | 通道启动程序跟踪是否应自动启动。 |
| ChinitTraceTableSize | 通道启动程序的跟踪数据空间的大小。 |
| ClusterSenderMonitoringDefault | 集群发送方通道的联机监视数据缺省值 |
| ClusterSender 统计信息 | 控制集群发送方通道的统计信息监视信息的收集。 |
| ClusterWorkloadData | 集群工作负载出口的用户数据 |
| ClusterWorkloadExit | 用于集群工作负载管理的用户出口的名称 |
| ClusterWorkloadLength | 传递到集群工作负载出口的消息数据的最大长度 |
| CLWLMRUChannels | 最近用于集群工作负载均衡的通道数 |
| CLWLUseQ | 集群工作负载使用远程队列。 |
| CodedCharSetId | 编码字符集标识 |
| CommandEvent | 命令事件的控制属性。 |
| CommandInputQName 属性 | 命令输入队列名称 |
| CommandLevel | 命令级别 |
| CommandServer"控制" 属性 | 命令服务器的控制属性。 |
| "配置事件" 属性 | 配置事件的控制属性。 |
| DeadLetterQName | 死信队列的名称 |
| DEFCLXQ | 缺省集群传输队列类型 |
| DefXmitQName | 缺省传输队列名称 |
| DistLists | 分发表支持 |
| DNSGroup | 使用工作负载管理器动态域名服务支持时 TCP 侦听器的组名。 |
| DNSWLM | TCP 侦听器是否向 Dynamic Domain Name Services 的工作负载管理器注册。 |
| ExpiryInterval | 扫描到期消息之间的时间间隔 |
| IGQPutAuthority | 组内排队放置权限 |
| IGQUserId | 组内排队用户标识 |
| InhibitEvent | 控制是否生成禁止 (禁止获取和禁止放入) 事件 |
| IPAddressVersion | Internet Protocol 地址的版本 |
| IntraGroupQueuing | 组内排队支持 |
| ListenerTimer | APPC 或 TCP/IP 故障后尝试重新启动侦听器之间的时间间隔。 |
| LocalEvent | 控制是否生成本地错误事件 |

表 572: 队列管理器的属性。

具有链接和简短描述的队列管理器属性的列表
(继续)

| 属性 | 描述 |
|--|---|
| LoggerEvent | 控制是否生成记录器事件 |
| LUGroupName | 用于处理队列共享组的进站传输的 LU 6.2 侦听器的通用 LU 名。 |
| LUName | 要用于出站 LU 6.2 传输的 LU 的名称。 |
| LU62ARMSuffix | SYS1.PARMLIB 成员 APPCPMxx, 用于指定此通道启动程序的 LUADD。 |
| LU62Channels | 使用 LU 6.2 的当前通道或已连接客户机的最大数目。 |
| MaxActiveChannels | 可以随时处于活动状态的最大通道数。 |
| MaxChannels | 当前通道的最大数目。 |
| MaxHandles | 最大句柄数 |
| MaxMsgLength | 最大消息长度 (字节) |
| MaxPriority 属性 | 最高优先级 |
| MaxPropertiesLength | 属性数据的最大长度 (以字节为单位) |
| MaxUncommittedMsgs | 工作单元中未落实的最大消息数 |
| MQIAccounting | 控制 MQI 数据的记帐信息收集。 |
| MQIStatistics | 控制队列管理器的统计信息监视信息的收集。 |
| MsgMarkBrowseInterval | 队列管理器可从浏览的消息中除去标记的时间间隔。 |
| OutboundPortMin | 使用 <i>OutboundPortMin</i> , 定义绑定出局通道时要使用的端口号范围。 |
| OutboundPortMax | 使用 <i>OutboundPortMax</i> , 定义绑定出局通道时要使用的端口号范围。 |
| PerformanceEvent | 控制是否生成与性能相关的事件 |
| 平台 | 运行队列管理器的平台 |
| PubSubNPInputMsg | 是废弃 (还是保留) 未传递的输入消息 |
| PubSubNPResponse | 控制未交付的行为 |
| PubSubMaxMsgRetryCount | 处理 (在同步点下) 失败的命令消息时的重试次数 |
| PubSubSyncPoint | 是否仅应在同步点下处理持久消息 (或处理所有消息) |
| PubSubMode | 排队的发布/预订接口是否正在运行 |
| QMGrDesc | 队列管理器描述 |
| QMGrIdentifier | 队列管理器的唯一内部生成的标识 |
| QMGrName | 队列管理器名称 |
| QSGName | 队列共享组的名称 |
| QueueAccounting | 控制队列的记帐信息收集。 |
| QueueMonitoring | 队列的联机监视数据 |
| QueueStatistics | 控制队列的统计数据收集。 |
| ReceiveTimeout | TCP/IP 通道在返回到不活动状态之前等待数据的时间长度。 |
| ReceiveTimeoutMin | <i>ReceiveTimeout</i> 的限定符。 |
| ReceiveTimeoutType | TCP/IP 通道在返回到不活动状态之前等待数据的最短时间。 |
| RemoteEvent | 控制是否生成远程错误事件 |
| RepositoryName | 此队列管理器为其提供存储库服务的集群的名称 |
| RepositoryNameList | 名称列表对象的名称, 其中包含此队列管理器为其提供存储库服务的集群的名称 |
| ScyCase | 安全概要文件的情况 |
| SharedQMGr 名称 | 共享队列管理器名称 |
| 第 728 页的『SPLCAP』 | 已开启或关闭队列管理器的 WebSphere MQ Advanced 消息安全保护。 |
| SSLCRLNameList 1 | 包含认证信息对象的名称的名称列表对象的名称。 |

| 表 572: 队列管理器的属性. | |
|---|------------------------------|
| 具有链接和简短描述的队列管理器属性的列表 (继续) | |
| 属性 | 描述 |
| SSLCryptoHardware 1 | 加密硬件配置字符串。 |
| SSLEvent | SSL 事件的控制属性。 |
| SSLFIPSRequired | 仅将 FIPS 认证的算法用于密码术。 |
| SSLKeyRepository 1 | SSL 密钥存储库的位置。 |
| SSLKeyReset 计数 | SSL 密钥重置计数。 |
| SSLTasks 1 | 用于处理 SSL 调用的服务器子任务数。 |
| StatisticsInterval | 写入统计信息监视数据的频率。 |
| StartStopEvent | 控制是否生成启动和停止事件 |
| SyncPoint | 同步点的可用性 |
| TCPChannels | 使用 TCP/IP 的当前通道或已连接客户机的最大数目。 |
| TCPKeepAlive | 是否使用 TCP KEEPALIVE 来检查其他连接端。 |
| TCPName | 您正在使用的 TCP/IP 系统的名称。 |
| TCPStackType | 通道启动程序如何使用 TCP/IP 地址。 |
| TraceRoute"记录" 属性 | 控制跟踪路由信息的记录。 |
| TriggerInterval | 触发器-消息时间间隔 |
| 版本 | 版本 |
| XrCapability | 指定是否支持 Telemetry 命令。 |
| 注意: | |
| 1. 无法使用 MQINQ 调用来查询此属性, 此部分中未描述此属性。请参阅 更改队列管理器 以获取此属性的详细信息。 | |

相关任务

指定运行时在 MQI 客户机上仅使用经过 FIPS 认证的 CipherSpecs

相关参考

适用于 UNIX, Linux 和 Windows 的联邦信息处理标准 (FIPS)

AccountingConn 覆盖 (MQLONG)

这允许应用程序覆盖 Qmgr 属性中 ACCTMQI 和 ACCTQDATA 值的设置。

该值为下列其中之一:

MQMON_DISABLED

应用程序无法使用 MQCONN 调用上 MQCNO 结构中的 "选项" 字段来覆盖 ACCTMQI 和 ACCTQ Qmgr 属性的设置。这是缺省值。

MQMON_ENABLED

应用程序可以使用 MQCNO 结构中的 "选项" 字段来覆盖 ACCTQ 和 ACCTMQI Qmgr 属性。

对该值的更改仅对在对属性进行更改之后与队列管理器的连接有效。

此属性仅在 IBM i, Unix 系统和 Windows 上受支持。

要确定此属性的值, 请将 MQIA_ACCOUNTING_CONN_OVERRIDE 选择器与 MQINQ 调用配合使用。

AccountingInterval (MQLONG)

这指定写入中间记帐记录之前的时间长度 (以秒计)。

该值是 0 到 604800 范围内的整数, 缺省值为 1800 (30 分钟)。指定 0 以关闭中间记录。

此属性仅在 IBM i, Windows, UNIX 和 Linux 系统上受支持。

要确定此属性的值, 请将 MQIA_ACCOUNTING_INTERVAL 选择器与 MQINQ 调用配合使用。

ActivityConn 覆盖 (MQLONG)

这允许应用程序覆盖队列管理器属性中 ACTVTRC 值的设置。

该值为下列其中之一：

MQMON_DISABLED

应用程序无法使用 MQCONN 调用上 MQCNO 结构中的 "选项" 字段来覆盖 ACTVTRC 队列管理器属性的设置。这是缺省值。

MQMON_ENABLED

应用程序可以使用 MQCNO 结构中的 "选项" 字段覆盖 ACTVTRC 队列管理器属性。

对该值的更改仅对在对属性进行更改之后与队列管理器的连接有效。

此属性仅在 IBM i, Unix 系统和 Windows 上受支持。

要确定此属性的值，请将 MQIA_ACTIVITY_CONN_OVERRIDE 选择器与 MQINQ 调用配合使用。

ActivityTrace (MQLONG)

这将控制 WebSphere MQ MQI 应用程序活动跟踪的收集。

该值为下列其中之一：

MQMON_ON

收集 WebSphere MQ MQI 应用程序活动跟踪。

MQMON_OFF

请勿收集 WebSphere MQ MQI 应用程序活动跟踪。这是缺省值。

如果将队列管理器属性 ACTVCON0 设置为 ENABLED，那么可能会使用 MQCNO 结构中的 "选项" 字段对各个连接覆盖此值。

对该值的更改仅对在对属性进行更改之后与队列管理器的连接有效。

此属性仅在 IBM i, Unix 系统和 Windows 上受支持。

要确定此属性的值，请将 MQIA_ACTIVITY_TRACE 选择器与 MQINQ 调用配合使用。

AdoptNewMCACheck (MQLONG)

这将定义要检查的元素，以确定在检测到与已处于活动状态的 MCA 同名的新入站通道时是否采用 MCA

该值为下列其中之一：

MQADOPT_CHECK_Q_MGR_NAME

检查队列管理器名称。

MQADOPT_CHECK_NET_ADDR

请检查网络地址。

MQADOPT_CHECK_ALL

请检查队列管理器名称和网络地址。如果可能，请执行此检查以防止通道被关闭，不慎或恶意关闭。这是缺省值。

MQADOPT_CHECK_NONE

请勿检查任何元素。

对此属性的更改将在通道下次尝试采用通道时生效。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_ADOPTNEWMCA_CHECK 选择器与 MQINQ 调用配合使用。

AdoptNewMCAType (MQLONG)

指定当检测到与 AdoptNewMCACheck 属性匹配的新入站通道请求时，是否自动重新启动特定通道类型的 MCA 的孤立实例

它是下列其中一个值：

MQADOPT_TYPE_NO

不需要采用孤立通道实例。这是缺省值。

MQADOPT_TYPE_ALL

采用所有通道类型。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_ADOPTNEWMCA_TYPE 选择器与 MQINQ 调用配合使用。

AlterationDate (MQCHAR12)

这是上次更改定义的日期。日期的格式为 YYYY-MM-DD，用两个尾部空格填充，使长度为 12 个字节。

要确定此属性的值，请将 MQCA_ALTERATION_DATE 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_DATE_LENGTH 提供。

AlterationTime (MQCHAR8)

这是上次更改定义的时间。时间的格式为 HH.MM.SS。

要确定此属性的值，请将 MQCA_ALTERATION_TIME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_TIME_LENGTH 提供。

AuthorityEvent (MQLONG)

这将控制是否生成授权 (未授权) 事件。它是下列其中一个值:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息，请参阅 [事件监视](#)。

要确定此属性的值，请将 MQIA_AUTHORITY_EVENT 选择器与 MQINQ 调用配合使用。

BridgeEvent (MQLONG)

这指定是否生成 IMS 网桥事件。

该值为下列其中之一:

MQEVR_ENABLED

生成 IMS 网桥事件，如下所示:

MQRC_BRIDGE_STARTED
MQRC_BRIDGE_STOPPED

MQEVR_DISABLED

请勿生成 IMS 网桥事件; 这是缺省值。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_BRIDGE_EVENT 选择器与 MQINQ 调用配合使用。

ChannelAutoDef (MQLONG)

此属性控制类型为 MQCHT_RECEIVER 和 MQCHT_SVRCONN 的通道的自动定义。始终启用 MQCHT_CLUSSDR 通道的自动定义。该值为下列其中之一:

MQCHAD_DISABLED

通道自动定义已禁用。

MQCHAD_ENABLED

已启用通道自动定义。

此属性仅在 AIX, HP-UX, IBM i, Linux, Solaris 和 Windows 上受支持。

要确定此属性的值，请将 MQIA_CHANNEL_AUTO_DEF 选择器与 MQINQ 调用配合使用。

ChannelAutoDefEvent (MQLONG)

这将控制是否生成通道自动定义事件。它适用于类型为 MQCHT_RECEIVER，MQCHT_SVRCONN 和 MQCHT_CLUSSDR 的通道。该值为下列其中之一：

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息，请参阅 [事件监视](#)。

此属性仅在 AIX，HP-UX，IBM i，Linux，Solaris 和 Windows 上受支持。

要确定此属性的值，请将 MQIA_CHANNEL_AUTO_DEF_EVENT 选择器与 MQINQ 调用配合使用。

ChannelAutoDefExit (MQCHARn)

这是自动通道定义的用户出口的名称。如果此名称为非空白，并且 *ChannelAutoDef* 的值为 MQCHAD_ENABLED，那么每次队列管理器要创建通道定义时都会调用该出口。这适用于类型为 MQCHT_RECEIVER，MQCHT_SVRCONN 和 MQCHT_CLUSSDR 的通道。然后，该出口可以执行下列其中一项操作：

- 创建通道定义而不进行更改。
- 修改所创建的通道定义的属性。
- 完全禁止创建通道。

注：此属性的长度和值都特定于环境。有关此属性在各种环境中的值的详细信息，请参阅 [第 922 页的『MQCD-通道定义』](#) 中的 MQCD 结构简介。

此属性仅在 AIX，HP-UX，IBM i，Linux，Solaris，Windows 和 z/OS 上受支持。在 z/OS 上，它仅适用于集群发送方和集群接收方通道。

要确定此属性的值，请将 MQCA_CHANNEL_AUTO_DEF_EXIT 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_EXIT_NAME_LENGTH 提供。

ChannelEvent (MQLONG)

这指定是否生成通道事件。

它是下列其中一个值：

MQEVR_EXCEPTION

仅生成以下通道事件：

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED，带有以下 ReasonQualifiers:
 - MQRQ_CHANNEL_STOPPED_ERROR
 - MQRQ_CHANNEL_STOPPED_RETRY
 - MQRQ_CHANNEL_STOPPED_DISABLED
- MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED

生成所有通道事件。即，除了由 EXCEPTION 生成的事件之外，还会生成以下通道事件：

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED，带有以下 ReasonQualifier:

MQRQ_CHANNEL_STOPPED_OK

MQEVR_DISABLED

不生成通道事件; 这是缺省值。

要确定此属性的值, 请将 MQIA_CHANNEL_EVENT 选择器与 MQINQ 调用配合使用。

ChannelInitiator 控制 (MQLONG)

这指定在队列管理器启动时是否启动通道启动程序。

它是下列其中一个值:

MQSVC_CONTROL_MANUAL

不会自动启动通道启动程序。

MQSVC_CONTROL_Q_MGR

通道启动程序将在队列管理器启动时自动启动。

要确定此属性的值, 请将 MQIA_CHINIT_CONTROL 选择器与 MQINQ 调用配合使用。

ChannelMonitoring (MQLONG)

这将指定通道的联机监视数据。

该值为下列其中之一:

MQMON_NONE

对所有通道禁用通道监视数据收集, 而不考虑 MONCHL 通道属性的设置。这是缺省值。

MQMON_OFF

对在 MONCHL 通道属性中指定 QMGR 的通道关闭监视数据收集。

MQMON_LOW

通过在 MONCHL 通道属性中指定 QMGR 的通道的低数据收集比率打开监视数据收集。

MQMON_MEDIUM

通过在 MONCHL 通道属性中指定 QMGR 的通道的适度数据收集比率打开监视数据收集。

MQMON_HIGH

通过在 MONCHL 通道属性中指定 QMGR 的通道的高数据收集比率打开监视数据收集。

要确定此属性的值, 请将 MQIA_MONITORING_CHANNEL 选择器与 MQINQ 调用配合使用。

ChannelStatistics (MQLONG)

这将控制通道的统计数据收集。

该值为下列其中之一:

MQMON_NONE

对所有通道的通道统计信息禁用数据收集, 而不考虑 STATCHL 通道属性的设置。这是缺省值。

MQMON_OFF

关闭在 STATCHL 通道属性中指定 QMGR 的通道的统计数据收集。

MQMON_LOW

打开统计信息数据收集, 但在 STATCHL 通道属性中指定 QMGR 的通道的数据收集比率较低。

MQMON_MEDIUM

打开统计信息数据收集, 在 STATCHL 通道属性中指定 QMGR 的通道的数据收集比率适中。

MQMON_HIGH

打开统计信息数据收集, 在 STATCHL 通道属性中指定 QMGR 的通道的数据收集比率较高。

对于大多数系统, 建议您使用 MEDIUM。但是, 对于每秒处理大量消息的通道, 您可能希望通过选择 LOW 来降低采样级别。此外, 对于仅处理几条消息的通道, 并且对于最新信息很重要的通道, 您可能想要选择 HIGH。

此属性仅在 IBM i, UNIX 系统和 Windows 上受支持。

要确定此属性的值, 请将 MQIA_STATISTICS_CHANNEL 选择器与 MQINQ 调用配合使用。

ChinitAdapters (MQLONG)

这是用于处理 WebSphere MQ 调用的适配器子任务数。值必须为 0-9999，缺省值为 8。

适配器与分派器 (ChinitDispatchers 属性) 的比率应该约为 8 到 5。但是，如果只有几个通道，那么不必将此参数的值从缺省值中减小。您可以使用以下值: 对于测试系统，8 (缺省值); 对于生产系统，20。理想情况下，您应该有 20 个适配器，这使 WebSphere MQ 调用具有更大的并行性。这对持久消息很重要。对于非持久消息，更少的适配器可能更好。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_CHINIT_ADAPTERS 选择器与 MQINQ 调用配合使用。

ChinitDispatchers (MQLONG)

这是要用于通道启动程序的分派器数。值必须为 0-9999，缺省值为 5。

作为准则，允许一个分派器用于 50 个当前通道。但是，如果只有几个通道，那么不必将此属性的值从缺省值中减小。如果您正在使用 TCP/IP，那么用于 TCP/IP 通道的最大分派器数为 100，即使您在此处指定更大的值也是如此。您可以使用以下设置: 测试系统 5 (缺省值); 生产系统 20 (需要 20 个分派器来处理最多 1000 个活动通道)。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_CHINIT_DISPATCHER 选择器与 MQINQ 调用配合使用。

ChinitTraceAutoStart (MQLONG)

这指定是否自动启动通道启动程序跟踪。

该值为下列其中之一:

MQTRAXSTR_YES

自动启动通道启动程序跟踪。这是缺省值。

MQTRAXSTR_NO

请勿自动启动通道启动程序跟踪。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_CHINIT_TRACE_AUTO_START 选择器与 MQINQ 调用配合使用。

ChinitTraceTableSize (MQLONG)

这是通道启动程序的跟踪数据空间大小 (MB)。

该值必须在范围 0 到 2048 之间，缺省值为 2。

注: 每当使用大型 z/OS 数据空间时，请确保系统上有足够的辅助存储器来支持任何相关的 z/OS 页面调度活动。您可能还需要增加您的 SYS1.DUMP 数据集大小。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_CHINIT_TRACE_TABLE_SIZE 选择器与 MQINQ 调用配合使用。

ClusterSenderMonitoringDefault (MQLONG)

这指定要替换自动定义的集群发送方通道的 ChannelMonitoring 属性的值。

该值为下列其中之一:

MQMON_Q_MGR

从队列管理器 ChannelMonitoring 属性的设置继承联机监视数据的集合。这是缺省值。

MQMON_OFF

关闭对通道的监视

MQMON_LOW

除非 ChannelMonitoring 为 MQMON_NONE，否则将以较低的数据收集速率开启监视，并且对系统性能的影响最小。收集的数据不可能是最新的。

MQMON_MEDIUM

除非 *ChannelMonitoring* 为 MQMON_NONE，否则将以适中的数据收集速率开启监视，但对系统性能的影响有限。

MQMON_HIGH

除非 *ChannelMonitoring* 为 MQMON_NONE，否则将以较高的数据收集速率开启监视，这可能会影响系统性能。收集的数据是最新的可用数据。

要确定此属性的值，请将 MQIA_MONITORING_AUTO_CLUSSDR 选择器与 MQINQ 调用配合使用。

ClusterSender 统计信息 (MQLONG)

由于可以根据存储库中 CLUSRCVR 的定义自动定义集群发送方通道，因此无法使用 ALTER 通道来更改这些自动定义的集群发送方通道的 STATCHL 属性设置。对于这些通道，是否收集联机监视数据的决定基于此队列管理器属性的设置。

该值为下列其中之一：

MQMON_Q_MGR

自动定义的集群发送方通道的统计信息数据收集基于队列管理器属性 STATCHL 的值。这是缺省值。

MQMON_OFF

关闭自动定义的集群发送方通道的统计数据收集。

MQMON_LOW

以较低的数据收集比率为自动定义的集群发送方通道打开统计数据收集。

MQMON_MEDIUM

对数据收集比率适中的自动定义集群发送方通道开启统计信息数据收集。

MQMON_HIGH

打开自动定义的集群发送方通道的统计信息数据收集，数据收集比率较高。

对于大多数系统，我们建议使用 MEDIUM。但是，对于每秒处理大量消息的自动定义集群发送方通道，您可能希望通过选择 LOW 来降低采样级别。此外，对于仅处理几条消息的通道，并且对于最新信息很重要的通道，您可能想要选择 HIGH。

要确定此属性的值，请将 MQIA_STATISTICS_AUTO_CLUSSDR 选择器与 MQINQ 调用配合使用。

ClusterWorkload 数据 (MQCHAR32)

这是用户定义的 32 字节字符串，调用时传递到集群工作负载出口。如果没有要传递到出口的数据，那么字符串为空白。

此属性仅在 AIX，HP-UX，IBM i，Linux，Solaris，Windows 和 z/OS 上受支持。

要确定此属性的值，请将 MQCA_CLUSTER_WORKLOAD_DATA 选择器与 MQINQ 调用配合使用。

ClusterWorkload 出口 (MQCHARn)

这是用于集群工作负载管理的用户出口的名称。如果此名称不为空，那么每次将消息放入集群队列或将消息从一个集群发送方队列移至另一个集群发送方队列时都会调用该出口。然后，出口可以接受队列管理器选择的队列实例作为消息的目标，或者选择另一个队列实例。

注：此属性的长度和值都特定于环境。

此属性仅在 AIX，HP-UX，IBM i，Linux，Solaris，Windows 和 z/OS 上受支持。

要确定此属性的值，请将 MQCA_CLUSTER_WORKLOAD_EXIT 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_EXIT_NAME_LENGTH 提供。

ClusterWorkload 长度 (MQLONG)

这是传递到集群工作负载出口的消息数据的最大长度。传递到出口的数据的实际长度是以下值的最小值：

- 消息长度。
- 队列管理器的 *MaxMsgLength* 属性。
- *ClusterWorkloadLength* 属性。

此属性仅在 AIX, HP-UX, IBM i, Linux, Solaris, Windows 和 z/OS 上受支持。

要确定此属性的值, 请将 MQIA_CLUSTER_WORKLOAD_LENGTH 选择器与 MQINQ 调用配合使用。

CLWLMRUChannels (MQLONG)

这指定要考虑由集群工作负载选择算法使用的最近使用的集群通道的最大数目。

这是 1 到 999999999 范围内的值。

要确定此属性的值, 请将 MQIA_CLWL_MRU_CHANNELS 选择器与 MQINQ 调用配合使用。

CLWLUseQ (MQLONG)

这指定是否将远程队列用于集群工作负载。

该值为下列其中之一:

MQCLWL_USEQ_ANY

同时使用本地队列和远程队列。

MQCLWL_USEQ_LOCAL

请勿使用远程队列。这是缺省值。

要确定此属性的值, 请将 MQIA_CLWL_USEQ 选择器与 MQINQ 调用配合使用。

CodedCharSetId (MQLONG)

这将定义队列管理器用于 MQI 中定义的所有字符串字段的字符集, 例如对象的名称以及队列创建日期和时间。字符集必须是对对象名中有效的字符具有单字节字符的字符集。它不适用于消息中携带的应用程序数据。该值取决于环境:

- 在 z/OS 上, 启动队列管理器时根据系统参数设置值; 缺省值为 500。
- 在 Windows 上, 该值是创建队列管理器的用户的主 CODEPAGE。
- 在 IBM i 上, 该值是首次创建队列管理器时在环境中设置的值。
- 在 UNIX 系统上, 该值是创建队列管理器的用户的语言环境的缺省 CODESET。

要确定此属性的值, 请将 MQIA_CODED_CHAR_SET_ID 选择器与 MQINQ 调用配合使用。

CommandEvent (MQLONG)

指定是否生成命令事件, 如下所示:

MQEVR_DISABLED

不生成命令事件。这是缺省值。

MQEVR_ENABLED

生成命令事件。

MQEVR_NO_DISPLAY

将为除 MQINQ 以外的所有成功命令生成命令事件。

要确定此属性的值, 请将 MQIA_COMMAND_EVENT 选择器与 MQINQ 调用配合使用。

CommandInputQName (MQCHAR48)

这是在本地队列管理器上定义的命令输入队列的名称。这是用户可以向其发送命令 (如果已授权) 的队列。队列的名称取决于环境:

- 在 z/OS 上, 队列的名称为 SYSTEM.COMMAND.INPUT; 可以向其发送 MQSC 和 PCF 命令。请参阅 [MQSC 命令](#) 以获取 MQSC 命令的详细信息, 并参阅 [可编程命令格式的定义](#) 以获取 PCF 命令的详细信息。
- 在所有其他环境中, 队列的名称为 SYSTEM.ADMIN.COMMAND.QUEUE, 并且只能向其发送 PCF 命令。但是, 如果 MQSC 命令包含在类型为 MQCMD_ESCAPE 的 PCF 命令中, 那么可以将 MQSC 命令发送到此队列。有关 Escape 命令的信息, 请参阅 [Escape](#)。

要确定此属性的值，请将 MQCA_COMMAND_INPUT_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

CommandLevel (MQLONG)

这指示队列管理器支持的系统控制命令的级别。这可以为以下值之一：

MQCMDL_LEVEL_1

系统控制命令的级别 1。

此值由以下版本的 IBM WebSphere MQ 返回：

- MQSeries for AIX V 2 发行版 2
- MQSeries
 - V 1 Release 1.1
 - 版本 1 发行版 1.2
 - 版本 1 发行版 1.3
- MQSeries for OS/400
 - 版本 2 发行版 3
 - 版本 3 发行版 1
 - 版本 3 发行版 6
- MQSeries for Windows V 2 发行版 0

MQCMDL_LEVEL_101

MQSeries for Windows V 2 Release 0.1。

MQCMDL_LEVEL_110

MQSeries for Windows V 2 Release 1。

MQCMDL_LEVEL_114

MQSeries for V 1 Release 1.4。

MQCMDL_LEVEL_120

MQSeries for V 1 Release 2.0。

MQCMDL_LEVEL_200

MQSeries for Windows NT V 2 Release 0。

MQCMDL_LEVEL_210

MQSeries for OS/390 V 2 Release 1.0。

MQCMDL_LEVEL_220

系统控制命令的级别 220。

此值由以下版本的 IBM WebSphere MQ 返回：

- MQSeries for AT&T GIS UNIX Version 2 Release 2
- MQSeries for SINIX and DC/OSx Version 2 Release 2
- MQSeries for SunOS V 2 发行版 2
- MQSeries for Tandem NonStop 内核版本 2 发行版 2

MQCMDL_LEVEL_221

系统控制命令的级别 221。

此值由 MQSeries for AIX V 2 Release 2.1 返回

MQCMDL_LEVEL_320

系统控制命令的级别 320。

此值由以下版本的 IBM WebSphere MQ 返回：

- MQSeries for OS/400
 - 版本 3 发行版 2

- 版本 3 发行版 7

MQCMDL_LEVEL_420

系统控制命令的级别 420。

此值由以下版本的 IBM WebSphere MQ 返回:

- IBM i 的 MQSeries
 - V 4 Release 2.0
 - 版本 4 发行版 2.1

MQCMDL_LEVEL_500

系统控制命令的级别 500。

此值由以下版本的 IBM WebSphere MQ 返回:

- IBM WebSphere MQ for AIX V 5 发行版 0
- MQSeries for HP-UX V 5 发行版 0
- MQSeries for Solaris V 5 发行版 0
- MQSeries for Windows NT V 5 发行版 0

MQCMDL_LEVEL_510

系统控制命令的级别 510。

此值由以下版本的 IBM WebSphere MQ 返回:

- IBM WebSphere MQ for AIX V 5 发行版 1
- MQSeries for AS/400 版本 5 发行版 1
- MQSeries for HP-UX V 5 发行版 1
- IBM WebSphere MQ for HP Integrity NonStop Server V 5 发行版 3
- MQSeries for Compaq Tru64 UNIX V 5 发行版 1
- MQSeries for Solaris V 5 发行版 1
- MQSeries for Windows NT V 5 发行版 1

MQCMDL_LEVEL_520

系统控制命令的级别 520。

此值由以下版本的 IBM WebSphere MQ 返回:

- MQSeries for AIX ersion 5 Release 2
- MQSeries for AS/400 版本 5 发行版 2
- MQSeries for HP-UX V 5 发行版 2
- MQSeries for Linux V 5 发行版 2
- MQSeries for OS/390 V 5 发行版 2
- MQSeries for Sun Solaris V 5 发行版 2
- MQSeries for Windows NT V 5 发行版 2

MQCMDL_LEVEL_530

系统控制命令的级别 530。

此值由以下版本的 IBM WebSphere MQ 返回:

- IBM WebSphere MQ for AIX V5.3
- IBM WebSphere MQ for HP-UX V5.3
- IBM WebSphere MQ for i/Series V 5 发行版 3
- IBM WebSphere MQ for Linux for Intel V 5 发行版 3
- IBM WebSphere MQ for Linux for zSeries V 5 发行版 3
- IBM WebSphere MQ for Solaris V5.3

- IBM WebSphere MQ for Windows V5.3
- IBM WebSphere MQ for z/OS V5.3

MQCMDL_LEVEL_600

系统控制命令的级别 600。

此值由以下版本的 IBM WebSphere MQ 返回:

- IBM WebSphere MQ for AIX Version 6.0
- IBM WebSphere MQ for HP-UX Version 6.0
- IBM WebSphere MQ for 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

系统控制命令的级别 700。

此值由以下版本的 IBM WebSphere MQ 返回:

- 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

系统控制命令的级别 701。

此值由以下版本的 IBM WebSphere MQ 返回:

- 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

系统控制命令的级别 710。

此值由以下版本的 IBM WebSphere MQ 返回:

- 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

750 级系统控制命令。

此值由以下版本的 IBM WebSphere MQ 返回:

- 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

对应于 *CommandLevel* 属性的特定值的系统控制命令集因 *Platform* 属性的值而异; 这两个命令都必须用于决定哪些系统控制命令受支持。

要确定此属性的值, 请将 MQIA_COMMAND_LEVEL 选择器与 MQINQ 调用配合使用。

CommandServer 控制 (MQLONG)

指定在队列管理器启动时是否启动命令服务器。

值可以是:

MQSVC_CONTROL_MANUAL

命令服务器不会自动启动。

MQSVC_CONTROL_Q_MGR

命令服务器将在队列管理器启动时自动启动。

此属性在 z/OS 上不受支持。

要确定此属性的值, 请将 MQIA_CMD_SERVER_CONTROL 选择器与 MQINQ 调用配合使用。

ConfigurationEvent (MQLONG)

控制是否生成配置事件。

要确定此属性的值, 请将 MQIA_CONFIGURATION_EVENT 选择器与 MQINQ 调用配合使用。

值可以是:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

DeadLetterQName (MQCHAR48)

这是在本地队列管理器上定义为死信 (undelivered-message) 队列的队列的名称。 如果无法将消息路由到其正确的目标, 那么会将这些消息发送到此队列。

例如, 在以下情况下, 消息将放入此队列中:

- 消息到达队列管理器, 其目标是尚未在该队列管理器上定义的队列
- 消息到达队列管理器, 但其目标队列无法接收消息, 原因可能是:
 - 队列已满
 - 禁止放置请求
 - 发送节点无权将消息放入队列

应用程序还可以将消息放在死信队列上。

报告消息的处理方式与普通消息相同; 如果报告消息无法传递到其目标队列 (通常是原始消息的消息描述符中的 *ReplyToQ* 字段指定的队列), 那么报告消息将放置在死信 (未传递的消息) 队列上。

注: 已超过到期时间 (请参阅 MQMD-到期字段) 的消息在被废弃时 **不会** 传输到此队列。但是, 如果发送应用程序请求, 那么仍会生成到期报告消息 (MQRO_EXPIRATION) 并将其发送到 *ReplyToQ* 队列。

如果发出放置请求的应用程序已通过 MQPUT 或 MQPUT1 调用返回的原因码 (例如, 在禁止放置请求的本地队列上放置的消息) 同步通知问题, 那么不会将消息放入死信 (不可传递的消息) 队列中。

死信 (undelivered-message) 队列上的消息有时会以 MQDLH 结构作为其应用程序消息数据的前缀。此结构包含额外的信息, 用于指示将消息放入死信 (undelivered-message) 队列的原因。请参阅 [第 307 页的『MQDLH-死信头』](#) 以获取此结构的更多详细信息。

此队列必须是本地队列, 并且 *Usage* 属性为 MQUS_NORMAL。

如果队列管理器不支持死信 (undelivered-message) 队列, 或者尚未定义一个队列, 那么名称全部为空白。所有 WebSphere MQ 队列管理器都支持死信 (undelivered-message) 队列, 但缺省情况下未定义此队列。

如果由于某种其他原因未定义, 完全或不可用死信 (undelivered-message) 队列, 那么将在传输队列上保留本应由消息通道代理传输到该队列的消息。

要确定此属性的值, 请将 MQCA_DEAD_LETTER_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

DefClusterXmitQueue 类型 (MQLONG)

DefClusterXmitQueue 类型 属性 控制缺省情况下集群发送方通道选择从哪个传输队列获取消息, 以将消息发送到集群接收方通道。

DefClusterXmitQueueType 的值为 MQCLXQ_SCTQ 或 MQCLXQ_CHANNEL。

MQCLXQ_SCTQ

所有集群发送方通道都从 SYSTEM.CLUSTER.TRANSMIT.QUEUE 发送消息。放置在传输队列上的消息的 *correlID* 将标识消息发往的集群发送方通道。

在定义队列管理器时会设置 SCTQ。此行为在低于 Version 7.5 的 IBM WebSphere MQ 版本中是隐式的。在较早版本中, 队列管理器属性 DefClusterXmitQueueType 不存在。

MQCLXQ_CHANNEL

每个集群发送方通道会从不同的传输队列发送消息。每个传输队列都创建为模型队列 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 中的永久动态队列。

此属性在 z/OS 上不受支持。

如果队列管理器属性 DefClusterXmitQueueType 设置为 CHANNEL, 缺省配置将更改为与各个集群传输队列相关联的集群发送方通道。传输队列是从模型队列 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 创建的永久动态队列。每个传输队列与一个集群发送方通道关联。当一个集群发送方通道为某个集群传输队列提供服务时, 传输队列仅包含用于一个集群中的一个队列管理器的消息。您可以配置集群, 以使一个集群中的每个队列管理器仅包含一个集群队列。在此情况下, 从一个队列管理器到每个集群队列的消息流量将独立于到其他队列的消息进行传输。

要查询该值, 请调用 MQINQ, 或者发送 "查询队列管理器" (MQCMD_INQUIRE_Q_MGR) PCF 命令, 请设置 MQIA_DEF_CLUSTER_XMIT_Q_TYPE 选择器。要更改该值, 请发送 "更改队列管理器" (MQCMD_CHANGE_Q_MGR) PCF 命令, 并设置 MQIA_DEF_CLUSTER_XMIT_Q_TYPE 选择器。

相关参考

[更改队列管理器](#)

[查询队列管理器](#)

[第 619 页的『MQINQ-查询对象属性』](#)

MQINQ 调用返回整数数组和一组包含对象属性的字符串。

DefXmitQName (MQCHAR48)

这是用于将消息传输到远程队列管理器的传输队列的名称 (如果没有其他指示要使用的传输队列)。

如果没有缺省传输队列, 那么名称完全为空白。此属性的初始值为空。

要确定此属性的值, 请将 MQCA_DEF_XMIT_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

DistLists (MQLONG)

这指示本地队列管理器是否支持 MQPUT 和 MQPUT1 调用上的分发列表。它是下列其中一个值:

MQDL_SUPPORTED

支持分发列表。

MQDL_NOT_SUPPORTED

不支持分发列表。

要确定此属性的值, 请将 MQIA_DIST_LISTS 选择器与 MQINQ 调用配合使用。

DNSGroup (MQCHAR18)

这是 TCP 侦听器的组名, 当使用 "工作负载管理器动态域名服务" 支持时, 该组将处理要加入的队列共享组的入站传输。最大长度为 18 个字符。如果将此名称留空, 那么将使用队列共享组名。

此属性仅在 z/OS 上受支持。

要确定此属性的值, 请将 MQCA_DNS_GROUP 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_DNS_GROUP_NAME_LENGTH 提供。

DNSWLM (MQLONG)

这指定处理队列共享组的入站传输的 TCP 侦听器是否向 Dynamic Domain Name Services 的工作负载管理器注册

该值为下列其中之一:

MQDNSWLM_YES

侦听器向工作负载管理器注册。

MQDNSWLM_NO

侦听器未向工作负载管理器注册。这是缺省值。

此属性仅在 z/OS 上受支持。

要确定此属性的值, 请将 MQIA_DNS_WLM 选择器与 MQINQ 调用配合使用。

ExpiryInterval (MQLONG)

这指示队列管理器扫描队列以查找到期消息的频率。它是 1 到 99 999 999 999 之间的时间间隔 (以秒为单位) 或以下特殊值:

MQEXPI_OFF

队列管理器不会扫描队列以查找到期消息。

要确定此属性的值, 请将 MQIA_EXPIRY_INTERVAL 选择器与 MQINQ 调用配合使用。

此属性仅在 z/OS 上受支持。

IGQPutAuthority (MQLONG)

仅当本地队列管理器是队列共享组的成员时, 此属性才适用。它指示当本地组内排队代理 (IGQ 代理) 从共享传输队列中除去消息并将该消息放入本地队列时执行的权限检查类型。该值为下列其中之一:

MQIGQPA_DEFAULT

针对授权检查的用户标识是单独 MQMD 中与消息关联的 *UserIdentifier* 字段的值 (当消息位于共享传输队列中时)。这是将消息放入共享传输队列的程序的标识, 通常与运行远程队列管理器的用户标识相同。

如果 RESLEVEL 概要文件指示要检查多个用户标识, 那么还会检查本地 IGQ 代理程序 (*IGQUserId*) 的用户标识。

MQIGQPA_CONTEXT

针对授权检查的用户标识是单独 MQMD 中与消息关联的 *UserIdentifier* 字段的值 (当消息位于共享传输队列中时)。这是将消息放入共享传输队列的程序的标识, 通常与运行远程队列管理器的用户标识相同。

如果 RESLEVEL 概要文件指示要检查多个用户标识，那么还将检查本地 IGQ 代理程序的用户标识 (*IGQUserId*) 以及嵌入式 MQMD 中 *UserIdentifier* 字段的值。后一个用户标识通常是发起消息的应用程序的用户标识。

MQIGQPA_ONLY_IGQ

检查授权的用户标识是本地 IGQ 代理程序 (*IGQUserId*) 的用户标识。

如果 RESLEVEL 概要文件指示要检查多个用户标识，那么此用户标识将用于所有检查。

MQIGQPA_ALTERNATE_OR_IGQ

检查授权的用户标识是本地 IGQ 代理程序 (*IGQUserId*) 的用户标识。

如果 RESLEVEL 概要文件指示要检查多个用户标识，那么还会检查 嵌入式 MQMD 中 *UserIdentifier* 字段的值。此用户标识通常是发起消息的应用程序的用户标识。

要确定此属性的值，请将 MQIA_IGQ_PUT_AUTHORITY 选择器与 MQINQ 调用配合使用。

此属性仅在 z/OS 上受支持。

IGQUserId (MQLONG)

仅当本地队列管理器是队列共享组的成员时，此属性才使用。它指定与本地组内排队代理 (IGQ 代理) 相关联的用户标识。此标识是当 IGQ 代理程序将消息放入本地队列时可以检查以获取授权的用户标识之一。检查的实际用户标识取决于 *IGQPutAuthority* 属性的设置以及外部安全性选项。

如果 *IGQUserId* 为空，那么不会将任何用户标识与 IGQ 代理程序相关联，并且不会执行相应的授权检查 (尽管仍可能检查其他用户标识以获取授权)。

要确定此属性的值，请将 MQCA_IGQ_USER_ID 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_USER_ID_LENGTH 给出。

此属性仅在 z/OS 上受支持。

InhibitEvent (MQLONG)

这将控制是否生成禁止 (禁止获取和禁止放入) 事件。该值为下列其中之一：

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息，请参阅 [事件监视](#)。

要确定此属性的值，请将 MQIA_处禁止事件选择器与 MQINQ 调用配合使用。

在 z/OS 上，无法使用 MQINQ 调用来确定此属性的值。

IntraGroup 排队 (MQLONG)

仅当本地队列管理器是队列共享组的成员时，此属性才适用。它指示是否对队列共享组启用组内排队。该值为下列其中之一：

MQIGQ_DISABLED

所有发往队列共享组中其他队列管理器的消息都是使用传统通道传输的。

MQIGQ_ENABLED

如果满足以下条件，那么将使用共享传输队列来传输发往队列共享组中其他队列管理器的消息：

- 消息数据和传输头的长度不超过 63 KB (64 512 字节)。

建议为传输头分配比 MQXQH 大小略多的空间；为此目的提供了常量 MQ_MSG_HEADER_LENGTH。

如果不满足此条件，那么将使用传统通道来传输消息。

注：当启用组内排队时，相对于使用传统通道传输的消息，不保留使用共享传输队列传输的消息的顺序。

要确定此属性的值，请将 MQIA_INTRA_GROUP_QUEUESELECTOR 与 MQINQ 调用配合使用。

此属性仅在 z/OS 上受支持。

IPAddressVersion (MQLONG)

指定使用的 IP 地址版本 (IPv4 或 IPv6)。

此属性仅与同时运行 IPv4 和 IPv6 的系统相关, 仅当满足下列其中一个条件时, 此属性才会影响定义为具有 *TransportType* MQXPY_TCP 的通道:

- 通道的 *ConnectionName* 是解析为 IPv4 和 IPv6 地址的主机名, 并且未指定其 *LocalAddress* 参数。
- 通道的 *ConnectionName* 和 *LocalAddress* 都是解析为 IPv4 和 IPv6 地址的主机名。

值可以是:

MQIPADDR_IPV4

使用 IPv4。

MQIPADDR_IPV6

使用 IPv6。

要确定此属性的值, 请将 MQIA_IP_ADDRESS_VERSION 选择器与 MQINQ 调用配合使用。

ListenerTimer (MQLONG)

这是 WebSphere MQ 尝试重新启动侦听器 (如果发生 APPC 或 TCP/IP 故障) 之间的时间间隔 (以秒计)。该值必须介于 5 和 9999 之间, 缺省值为 60。

此属性仅在 z/OS 上受支持。

要确定此属性的值, 请将 MQIA_LISTENER_TIMER 选择器与 MQINQ 调用配合使用。

LocalEvent (MQLONG)

这将控制是否生成本地错误事件。该值为下列其中之一:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息, 请参阅 [事件监视](#)。

要确定此属性的值, 请将 MQIA_LOCAL_EVENT 选择器与 MQINQ 调用配合使用。

在 z/OS 上, 无法使用 MQINQ 调用来确定此属性的值。

LoggerEvent (MQLONG)

这将控制是否生成恢复日志事件。该值为下列其中之一:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息, 请参阅 [事件监视](#)。

要确定此属性的值, 请将 MQIA_LOGGER_EVENT 选择器与 MQINQ 调用配合使用。

此属性仅在 AIX, HP-UX, IBM i, Linux, Solaris 和 Windows 上受支持。

LUGroupName (MQCHAR8)

这是用于处理队列共享组的进站传输的 LU 6.2 侦听器的通用 LU 名。如果将此名称留空, 那么不能使用此侦听器。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQCA_LU_GROUP_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_LU_NAME_LENGTH 指定。

LUName (MQCHAR8)

这是要用于出站 LU 6.2 传输的 LU 的名称。将此设置为侦听器用于入站传输的 LU。如果将此名称留空，那么将使用 APPC/MVS 缺省 LU；这是变量，因此如果使用的是 LU6.2，请始终设置 LUName。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQCA_LU_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_LU_NAME_LENGTH 指定。

LU62ARMSuffix (MQCHAR2)

这是 SYS1.PARMLIB 成员 APPCPMxx，用于指定此通道启动程序的 LUADD。当 ARM 重新启动通道启动程序时，将发出 z/OS 命令 SET APPC=xx。如果保留此名称为空，那么不会发出 SET APPC=xx。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQCA_LU62_ARM_SUFFIX 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_ARM_SUFFIX_LENGTH 指定。

LU62Channels (MQLONG)

这是使用 LU 6.2 传输协议的当前通道或可连接的客户端的最大通道数。

该值必须在范围 0 到 9999 之间，缺省值为 200。如果将此值设置为零，那么不会使用 LU 6.2 传输协议。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_LU62_CHANNELS 选择器与 MQINQ 调用配合使用。

MaxActive 通道 (MQLONG)

此属性是可以随时处于活动状态的最大通道数。

缺省值是为 MaxChannels 属性指定的值。对于 z/OS，该值必须在 1 到 9 999 范围内。对于所有其他平台，该值必须在 1 到 65535 范围内。

要确定此属性的值，请将 MQIA_ACTIVE_CHANNELS 选择器与 MQINQ 调用配合使用。

相关概念

[通道状态](#)

MaxChannels (MQLONG)

此属性是可以是当前的最大通道数 (包括具有已连接客户端的服务器连接通道)。

对于 z/OS，该值必须在 1 到 9 999 范围内，缺省值为 200。对于所有其他平台，该值必须在 1 到 65535 范围内，缺省值为 100。忙于为来自网络的连接提供服务的系统可能需要比缺省设置更高的数字。确定适合您环境的值，理想情况下是通过在测试期间观察系统的行为。

对于 z/OS 以外的平台，MaxChannels 的值在相应队列管理器的 qm.ini 文件中设置。

要确定此属性的值，请将 MQIA_MAX_CHANNELS 选择器与 MQINQ 调用配合使用。

相关概念

[通道状态](#)

MaxHandles (MQLONG)

这是任何一个任务可以同时使用的最大打开句柄数。针对单个队列 (或针对不是队列的对象) 的每个成功 MQOPEN 调用都使用一个句柄。该句柄在对象关闭时可供复用。但是，当打开分发列表时，将为分发列表中的每个队列分配一个单独的句柄，因此 MQOPEN 调用将使用与分发列表中有队列一样多的句柄。在决定 MaxHandles 的合适值时，必须考虑此问题。

MQPUT1 调用在其处理过程中执行 MQOPEN 调用; 因此, MQPUT1 使用的句柄数与 MQOPEN 使用的句柄数相同, 但这些句柄仅用于 MQPUT1 调用本身的持续时间。

在 z/OS 上, 任务表示 CICS 任务, MVS 任务或 IMS 从属区域。

该值在范围 1 到 999 999 999 之间。缺省值由环境确定:

- 在 z/OS 上, 缺省值为 100。
- 在所有其他环境中, 缺省值为 256。

要确定此属性的值, 请将 MQIA_MAX_句柄选择器与 MQINQ 调用配合使用。

MaxMsg 长度 (MQLONG)

这是队列管理器可以处理的最长物理消息的长度。但是, 由于可以独立于 *MaxMsgLength* 队列属性设置 *MaxMsgLength* 队列管理器属性, 因此可以放置在队列上的最长物理消息是这两个值中较小的值。

如果队列管理器支持分段, 那么应用程序可以放置比两个 *MaxMsgLength* 属性中的较小值长的逻辑消息, 但仅当应用程序在 MQMD 中指定了 MQMF_SEGMENTATION_ALLOWED 标志时才这样做。如果指定了该标志, 那么逻辑消息的长度上限为 999 999 999 字节, 但通常由操作系统或运行应用程序的环境施加的资源约束会导致下限。

MaxMsgLength 属性的下限为 32 KB (32 768 字节)。上限为 100 MB (104 857 600 字节)。

要确定此属性的值, 请将 MQIA_MAX_MSG_LENGTH 选择器与 MQINQ 调用配合使用。

MaxPriority (MQLONG)

这是队列管理器支持的最大消息优先级。优先级范围从零 (最低) 到 *MaxPriority* (最高)。

要确定此属性的值, 请将 MQIA_MAX_PRIORITY 选择器与 MQINQ 调用配合使用。

MaxProperties 长度 (MQLONG)

这用于控制可随消息一起流动的属性的属性的大小。这包括属性名称 (以字节计) 和属性值的大小 (以字节计)。

要确定此属性的值, 请将 MQIA_MAX_PROPERTIES_LENGTH 选择器与 MQINQ 调用配合使用。

MaxUncommitted 消息 (MQLONG)

这是工作单元中可以存在的最大未落实消息数。未落实消息数是自当前工作单元启动以来的以下消息数的总和:

- 应用程序使用 MQPMO_SYNCPOINT 选项放置的消息
- 应用程序使用 MQGMO_SYNCPOINT 选项检索的消息
- 由队列管理器针对使用 MQPMO_SYNCPOINT 选项放置的消息生成的触发器消息和 COA 报告消息
- 由队列管理器针对使用 MQGMO_SYNCPOINT 选项检索的消息生成的 COD 报告消息

以下内容 不 计为未落实的消息:

- 应用程序在工作单元外部放入或检索的消息
- 由于在工作单元外部放置或检索消息而触发消息或队列管理器生成的 COA/COD 报告消息
- 队列管理器生成的到期报告消息 (即使导致到期报告消息的调用指定了 MQGMO_SYNCPOINT)
- 队列管理器生成的事件消息 (即使引起事件消息的调用指定了 MQPMO_SYNCPOINT 或 MQGMO_SYNCPOINT)

注:

1. 异常报告消息由消息通道代理程序 (MCA) 或应用程序生成, 并以与应用程序放入或检索的普通消息相同的方式处理。
2. 当使用 MQPMO_SYNCPOINT 选项放置消息或段时, 未落实的消息数将增加 1, 而不考虑该放置实际产生的物理消息数。(如果队列管理器必须对消息或段进行细分, 那么可能会产生多条物理消息。)
3. 使用 MQPMO_SYNCPOINT 选项放置分发列表时, 未落实的消息数将增加一个 针对生成的每条物理消息。这可以是一个或多个小目标, 也可以是分发列表中的目标数。

此属性的下限为 1; 上限为 999 999。缺省值是 10000。

要确定此属性的值, 请将 MQIA_MAX_UNCOMMITTED_MSGS 选择器与 MQINQ 调用配合使用。

MQIAccounting (MQLONG)

这将控制 MQI 数据的记帐信息收集。

该值为下列其中之一:

MQMON_ON

收集 API 记帐数据。

MQMON_OFF

请勿收集 API 记帐数据。这是缺省值。

如果将队列管理器属性 ACCTCONO 设置为 ENABLED, 那么可能会使用 MQCNO 结构中的 "选项" 字段对各个连接覆盖此值。对此值所作的更改仅对在对属性进行更改之后与队列管理器进行的连接有效。

此属性仅在 IBM i, UNIX 系统和 Windows 上受支持。

要确定此属性的值, 请将 MQIA_ACCOUNTING_MQI 选择器与 MQINQ 调用配合使用。

MQIStatistics (MQLONG)

这控制队列管理器的统计信息监视信息收集。

该值为下列其中之一:

MQMON_ON

收集 MQI 统计信息。

MQMON_OFF

请勿收集 MQI 统计信息。这是缺省值。

此属性仅在 IBM i, UNIX and Linux 系统和 Windows 上受支持。

要确定此属性的值, 请将 MQIA_STATISTICS_MQI 选择器与 MQINQ 调用配合使用。

MsgMarkBrowseInterval (MQLONG)

时间间隔 (以毫秒为单位), 在此时间间隔之后, 队列管理器可以自动从浏览消息中除去标记。

这是一个时间间隔 (以毫秒计), 在此时间间隔之后, 队列管理器可以自动从浏览消息中除去标记。

此属性描述通过调用 MQGET (使用 get 消息选项 MQGMO_MARK_BROWSE_CO_OP) 将消息标记为已浏览的时间间隔。

队列管理器可能会自动取消标记已被标记为已被协作句柄集合浏览的已浏览消息 (如果这些消息已被标记超过此近似时间间隔)。

这不会影响通过调用 MQGET (使用 get 消息选项 MQGMO_MARK_BROWSE_HANDLE) 获取的任何标记为浏览的消息的状态。

最大值为 999 999 999, 缺省值为 5000。MsgMarkBrowseInterval 的特殊值 -1 表示不受限制的时间间隔。



注意: 此值不应低于缺省值 5000。

要确定此属性的值, 请将 MQIA_MSG_MARK_BROWSE_INTERVAL 选择器与 MQINQ 调用配合使用。

OutboundPort 最大 (MQLONG)

这是要用于绑定传出通道的端口号的范围内的最高端口号 (由 OutboundPortMin 和 OutboundPortMax 定义)。

该值是 0 到 65535 范围内的整数, 必须等于或大于 OutboundPort 的最小值。缺省值为 0。

此属性仅在 z/OS 上受支持。

要确定此属性的值, 请将 MQIA_OUTBOUND_PORT_MAX 选择器与 MQINQ 调用配合使用。

OutboundPort 最小值 (MQLONG)

这是要用于绑定传出通道的端口号范围内的最低端口号 (由 OutboundPortMin 和 OutboundPortMax 定义)。

该值是 0 到 65535 范围内的整数, 并且必须等于或小于 OutboundPort 最大值。缺省值为 0。

此属性仅在 z/OS 上受支持。

要确定此属性的值, 请将 MQIA_OUTBOUND_PORT_MIN 选择器与 MQINQ 调用配合使用。

PerformanceEvent (MQLONG)

这将控制是否生成与性能相关的事件。它是下列其中一个值:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息, 请参阅 [事件监视](#)。

要确定此属性的值, 请将 MQIA_PERFORMANCE_EVENT 选择器与 MQINQ 调用配合使用。

平台 (MQLONG)

这指示运行队列管理器的操作系统:

MQPL_AIX

AIX (与 MQPL_UNIX 的值相同)。

MQPL_MVS

z/OS (与 MQPL_ZOS 相同的值)。

MQPL_NSK

HP Integrity NonStop Server.

MQPL_OS390

z/OS (与 MQPL_ZOS 相同的值)。

MQPL_OS400

IBM i.

MQPL_UNIX

UNIX 系统。

MQPL_WINDOWS_NT

Windows 系统。

MQPL_ZOS

z/OS.

要确定此属性的值, 请将 MQIA_PLATFORM 选择器与 MQINQ 调用配合使用。

PubSubNPInputMsg (MQLONG)

是废弃还是保留未传递的输入消息。

该值为下列其中之一:

MQUNDELIVERED_DISCARD

如果不能处理非持久输入消息, 那么会废弃这些消息。

这是缺省值。

MQUNDELIVERED_KEEP

如果不能处理非持久输入消息, 将不会废弃这些消息。在此情况下, 排队的发布/预订接口将继续以适当的时间间隔重试该进程, 并且不会继续处理后续消息。

要确定此属性的值, 请将 MQIA_PUBSUB_NP_MSG 选择器与 MQINQ 调用配合使用。

PubSubNPResponse (MQLONG)

控制未传递的响应消息的行为。

该值为下列其中之一：

MQUNDELIVERED_NORMAL

不能放在应答队列上的非持久响应将放在死信队列上，如果不能放在 DLQ 上，那么将废弃这些非持久响应。

MQUNDELIVERED_SAFE

不能放在应答队列上的非持久响应将放在死信队列上。 如果无法设置响应并且无法将其放在 DLQ 上，那么排队的发布/预订接口将回滚当前操作，然后以适当的时间间隔重试，并且不会继续处理后续消息。

MQUNDELIVERED_DISCARD

不会将非持久响应放置在应答队列上。

这是新队列管理器的缺省值。

MQUNDELIVERED_KEEP

非持久响应不会放在死信队列上或被废弃。 相反，排队的发布/预订接口将回退当前操作，然后以适当的时间间隔重试该操作。

要确定此属性的值，请将 MQIA_PUBSUB_NP_RESP 选择器与 MQINQ 调用配合使用。

已迁移队列管理器的缺省值。

如果已从 WebSphere MQ V6.0 迁移队列管理器，那么此属性的初始值取决于迁移前 DiscardNonPersistentResponse 和 DLQNonPersistent 响应的值，如下表所示。

| | | DLQNonPersistent 响应 | | |
|------------------------------|-----|---|--|---|
| | | Yes | 否 | 未设定 |
| DiscardNonPersistentResponse | 是 | MQUNDELIVERED_NORMAL | MQUNDELIVERED_DISCARD | MQUNDELIVERED_NORMAL |
| | 否 | MQUNDELIVERED_SAFE | MQUNDELIVERED_KEEP | MQUNDELIVERED_SAFE |
| | 未设置 | 如果 SyncPointPersistent = No , 那么 MQUNDELIVERED_SAFE else MQUNDELIVERED_NORMAL | 如果 SyncPointPersistent = No , 那么 MQUNDELIVERED_KEEP else MQUNDELIVERED_DISCARD | 如果 SyncPointPersistent = No , 那么 MQUNDELIVERED_SAFE else MQUNDELIVERED_NORMAL |

PubSubMaxMsgRetryCount (MQLONG)

在同步点下处理失败命令消息时的重试次数。

该值为下列其中之一：

0 - 999 999 999

缺省值是 5。

要确定此属性的值，请将 MQIA_PUBSUB_MAXMSG_RETRY_COUNT 选择器与 MQINQ 调用配合使用。

PubSubSyncPoint (MQLONG)

是仅在同步点下处理持久消息还是所有消息。

该值为下列其中之一：

MQSYNCPOINT_IFPER

这将使排队的发布/预订接口在同步点外部接收非持久消息。 如果守护程序接收到同步点外部的发布，那么该守护程序会将发布转发至它所知道的 同步点以外的订户。

这是缺省值。

MQSYNCPOINT_YES

这将使排队的发布/预订接口接收同步点下的所有消息。

要确定此属性的值，请将 MQIA_PUBSUB_SYNC_PT 选择器与 MQINQ 调用配合使用。

PubSub 方式 (MQLONG)

发布/预订引擎和排队的发布/预订接口是否正在运行，因此允许应用程序使用应用程序编程接口以及排队的发布/预订接口所监视的队列来发布/预订。

该值为下列其中之一：

MQPSM_COMPAT

发布/预订引擎正在运行。因此，可以使用应用程序编程接口来发布/预订。已排队的发布/预订接口未在运行，因此不会对放入已排队的发布/预订接口所监视的队列的任何消息执行操作。此设置用于与使用此队列管理器的 WebSphere Message Broker V6 或更低版本兼容，因为它必须读取排队的发布/预订接口通常读取的相同队列。

MQPSM_DISABLED

发布/预订引擎和排队的发布/预订接口未在运行。因此，无法使用应用程序编程接口来发布/预订。不会对放入已排队的发布/预订接口所监视的队列的任何发布/预订消息执行操作。

MQPSM_ENABLED

发布/预订引擎和排队的发布/预订接口正在运行。因此，可以使用应用程序编程接口和由排队的发布/预订接口监视的队列来发布/预订。这是队列管理器的初始缺省值。

要确定此属性的值，请将 MQIA_PUBSUB_MODE 选择器与 MQINQ 调用配合使用。

QMGrDesc (MQCHAR64)

将此字段用于描述队列管理器的注释。该字段的内容对队列管理器没有意义，但队列管理器可能要求该字段仅包含可显示的字符。它不能包含任何空字符；如有必要，将空格填充到右边。在 DBCS 安装中，此字段可包含 DBCS 字符 (最大字段长度为 64 字节)。

注：如果此字段包含不在队列管理器的字符集中的字符 (如 *CodedCharSetId* 队列管理器属性所定义)，那么如果将此字段发送到另一个队列管理器，那么这些字符可能转换不正确。

- 在 z/OS 上，缺省值为产品名称和版本号。
- 在所有其他环境中，缺省值为空白。

要确定此属性的值，请将 MQCA_Q_MGR_DESC 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_MGR_DESC_LENGTH 提供。

QMGrIdentifier (MQCHAR48)

这是队列管理器的内部生成的唯一名称。

要确定此属性的值，请将 MQCA_Q_MGR_IDENTIFIER 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_MGR_IDENTIFIER_LENGTH 指定。

此属性在以下环境中受支持：AIX，HP-UX，z/OS，IBM i，Solaris，Linux，Windows 以及连接到这些系统的 WebSphere MQ 客户机。

QMGrName (MQCHAR48)

这是本地队列管理器的名称，即应用程序所连接的队列管理器的名称。

名称的前 12 个字符用于构造唯一消息标识 (请参阅 MQMD-MsgId 字段)。因此，可以相互通信的队列管理器必须具有在前 12 个字符中不同的名称，以便消息标识在队列管理器网络中是唯一的。

在 z/OS 上，该名称与子系统名称相同，限制为 4 个非空白字符。

要确定此属性的值，请将 MQCA_Q_MGR_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_MGR_NAME_LENGTH 指定。

QSGName (MQCHAR4)

这是本地队列管理器所属的队列共享组的名称。如果本地队列管理器不属于队列共享组，那么名称为空。

要确定此属性的值，请将 MQCA_QSG_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_QSG_NAME_LENGTH 提供。

此属性仅在 z/OS 上受支持。

QueueAccounting (MQLONG)

这将控制队列的记帐信息收集。

该值为下列其中之一：

MQMON_NONE

无论队列记帐属性 ACCTQ 的设置如何，都不要收集队列的记帐数据。这是缺省值。

MQMON_OFF

请勿收集在 ACCTQ 队列属性中指定 QMGR 的队列的记帐数据。

MQMON_ON

收集在 ACCTQ 队列属性中指定 QMGR 的队列的记帐数据。

对此值所作的更改仅在对属性进行更改之后与队列管理器进行的连接有效。

要确定此属性的值，请将 MQIA_ACCOUNTING_Q 选择器与 MQINQ 调用配合使用。

QueueMonitoring (MQLONG)

这指定用于对队列进行联机监视的缺省设置。

如果 *QueueMonitoring* 队列属性设置为 MQMON_Q_MGR，那么此属性指定通道假定的值。值可以是：

MQMON_OFF

已关闭联机监视数据收集。这是队列管理器的初始缺省值。

MQMON_NONE

将对队列关闭联机监视数据收集，而不考虑其 *QueueMonitoring* 属性的设置。

MQMON_LOW

开启在线监控数据采集，数据采集比例低。

MQMON_MEDIUM

开启在线监控数据采集，数据采集比例适中。

MQMON_HIGH

开启在线监控数据采集，数据采集比例高。

要确定此属性的值，请将 MQIA_MONITORING_Q 选择器与 MQINQ 调用配合使用。

QueueStatistics (MQLONG)

这将控制队列的统计信息数据收集。

它是下列其中一个值：

MQMON_NONE

请勿收集队列的队列统计信息，而不考虑 *QueueStatistics* 队列属性的设置。这是缺省值。

MQMON_OFF

请勿收集在 *QueueStatistics* 队列属性中指定队列管理器的队列的统计数据。

MQMON_ON

收集在 *QueueStatistics* 队列属性中指定队列管理器的队列的统计数据。

要确定此属性的值，请将 MQIA_STATISTICS_Q 选择器与 MQINQ 调用配合使用。

ReceiveTimeout (MQLONG)

这指定 TCP/IP 通道在返回到不活动状态之前等待从其伙伴接收数据 (包括脉动信号) 的时间长度。它仅适用于消息通道，而不适用于 MQI 通道。

ReceiveTimeout 的确切含义由 ReceiveTimeout 类型中指定的值改变。ReceiveTimeout 类型可以设置为下列其中一项：

- MQRCVTIME_EQUAL-此值是通道等待的秒数。指定 0-999999 范围内的值。
- MQRCVTIME_ADD-此值是要添加到协商的 HBINT 的秒数，它确定通道等待的时间长度。指定 1-999999 范围内的值。
- MQRCVTIME_MULTIPLY-此值是应用于协商的 HBINT 的乘数。指定值 0 或 2-99 范围内的值。

缺省值为 0。

将 `ReceiveTimeout` 类型设置为 `MQRCVTIME_MULTIPLY` 或 `MQRCVTIME_EQUAL`，并将 `ReceiveTimeout` 设置为 0，以阻止通道超时等待从其合作伙伴接收数据。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 `MQIA_RECEIVE_TIMEOUT` 选择器与 `MQINQ` 调用配合使用。

ReceiveTimeout 最小值 (MQLONG)

这是 TCP/IP 通道在返回到不活动状态之前等待从其伙伴接收数据 (包括脉动信号) 的最短时间 (以秒计)。

它仅适用于消息通道，而不适用于 MQI 通道。该值必须在 0 到 999999 的范围内，缺省值为 0。

如果使用 `ReceiveTimeout` 类型来指定将相对于 HBINT 的协商值计算 TCP/IP 通道等待时间，并且生成的值小于此参数的值，那么将改为使用此值。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 `MQIA_RECEIVE_TIMEOUT_MIN` 选择器与 `MQINQ` 调用配合使用。

ReceiveTimeout 类型 (MQLONG)

这是应用于 `ReceiveTimeout` 的限定符，用于定义 TCP/IP 通道在返回到不活动状态之前等待从其合作伙伴接收数据 (包括脉动信号) 的时间长度。它仅适用于消息通道，而不适用于 MQI 通道。

该值为下列其中之一：

MQRCVTIME_MULTIPLY

`ReceiveTimeout` 是应用于协商的 HBINT 值以确定通道等待时间的乘数。这是缺省值。

MQRCVTIME_ADD

`ReceiveTimeout` 是一个值 (以秒计)，用于添加到协商的 HBINT 值以确定通道等待的时间长度。

MQRCVTIME_EQUAL

`ReceiveTimeout` 是通道等待的值 (以秒计)。

要停止通道超时等待从其合作伙伴接收数据，请将 `ReceiveTimeout` 类型设置为 `MQRCVTIME_MULTIPLY` 或 `MQRCVTIME_EQUAL`，并将 `ReceiveTimeout` 设置为 0。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 `MQIA_RECEIVE_TIMEOUT_TYPE` 选择器与 `MQINQ` 调用配合使用。

RemoteEvent (MQLONG)

这将控制是否生成远程错误事件。它是下列其中一个值：

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息，请参阅 [事件监视](#)。

要确定此属性的值，请将 `MQIA_REMOTE_EVENT` 选择器与 `MQINQ` 调用配合使用。

RepositoryName (MQCHAR48)

这是此队列管理器为其提供存储库管理器服务的集群的名称。如果队列管理器为多个集群提供此服务，那么 `RepositoryNameList` 指定用于标识集群的名称列表对象的名称，并且 `RepositoryName` 为空白。`RepositoryName` 和 `RepositoryNameList` 中的至少一个必须为空白。

此属性仅在 AIX, HP-UX, IBM i, Linux, Solaris, Windows 和 z/OS 上受支持。

要确定此属性的值，请将 `MQCA_REPOSITORY_NAME` 选择器与 `MQINQ` 调用配合使用。此属性的长度由 `MQ_Q_MGR_NAME_LENGTH` 指定。

RepositoryNameList (MQCHAR48)

这是名称列表对象的名称，其中包含此队列管理器为其提供存储库管理器服务的集群的名称。如果队列管理器仅为一个集群提供此服务，那么名称列表对象仅包含一个名称。或者，可以使用 *RepositoryName* 来指定集群的名称，在这种情况下，*RepositoryNameList* 为空白。*RepositoryName* 和 *RepositoryNameList* 中的至少一个必须为空白。

此属性仅在 AIX, HP-UX, IBM i, Linux, Solaris, Windows 和 z/OS 上受支持。

要确定此属性的值，请将 MQCA_REPOSITORY_NAMELIST 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_NAMELIST_NAME_LENGTH 提供。

ScyCase(MQCHAR8)

指定队列管理器是支持大小写混合的安全概要文件名称，还是仅支持大写形式的安全概要文件名称。

该值为下列其中之一：

MQSCYC_UPPER

安全概要文件名称必须为大写。

MQSCYC_MIXED

安全概要文件名称可以是大写或混合大小写。

在指定了 *SecurityType* (*MQSECTYPE_CLASSES*) 的情况下运行 "刷新安全性" 命令时，对此属性的更改将生效。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_SECURITY_CASE 选择器与 MQINQ 调用配合使用。

SharedQMgr 名称 (MQLONG)

这指定对于共享队列，当 *ObjectQmgrName* 是队列共享组中另一个队列管理器的时，是否应将 *ObjectQmgrName* 用作 MQOPEN 调用上的本地队列管理器或将其视为本地队列管理器。

值可以是：

MQSQQM_USE

将使用 *ObjectQmgrName* 并打开相应的传输队列。

MQSQQM_IGNORE

如果共享目标队列，并且 *ObjectQmgrName* 是同一队列共享组中的队列管理器的队列管理器，那么将在本地执行打开操作。

此属性仅在 z/OS 上有效。

要确定此属性的值，请将 MQIA_SHARED_Q_Q_MGR_NAME 选择器与 MQINQ 调用配合使用。

SPLCAP

指示 WebSphere MQ Advanced Message Security 的安全功能是否可用于队列管理器。

MQCAP_SUPPORTED

如果为运行队列管理器的安装安装了 WebSphere MQ AMS 组件，那么这是缺省值。

MQCAP_NOT_SUPPORTED

SSLEvent (MQLONG)

这指定是否生成 SSL 事件。

它是下列其中一个值：

MQEVR_ENABLED

生成 SSL 事件，如下所示：

MQRC_CHANNEL_SSL_ERROR

MQEVR_DISABLED

不生成 SSL 事件；这是缺省值。

要确定此属性的值，请将 MQIA_SSL_EVENT 选择器与 MQINQ 调用配合使用。

SSLFIPSRequired (MQLONG)

这允许您指定仅当在 WebSphere MQ 中 (而不是在加密硬件中) 执行密码术时, 才使用 FIPS 认证的算法。如果配置了加密硬件, 那么所使用的加密模块是由硬件产品提供的模块; 根据所使用的硬件产品, 这些模块可能已通过 FIPS 认证, 也可能未通过 FIPS 认证到特定级别。

该值是下列其中一个值:

MQSSL_FIPS_NO

使用正在使用的平台上支持的任何 CipherSpec。该值为缺省值。

MQSSL_FIPS_YES

在来自此队列管理器的所有 SSL 连接上允许的 CipherSpecs 中仅使用 FIPS 认证的密码算法。

此参数仅在 UNIX, Linux, Windows 和 z/OS 平台上有效。

要确定此属性的值, 请将 MQIA_SSL_FIPS_REQUIRED 选择器与 MQINQ 调用配合使用。

相关任务

指定运行时在 MQI 客户机上仅使用经过 FIPS 认证的 CipherSpecs

相关参考

适用于 UNIX, Linux 和 Windows 的联邦信息处理标准 (FIPS)

SSLKeyReset 计数 (MQLONG)

这指定何时启动通信的 SSL 通道消息通道代理程序 (MCA) 重置用于通道上加密的密钥。

此值表示重新协商密钥之前在通道上发出和接收的未加密字节的总数。此字节数包括由 MCA 发送的控制信息。

该值是范围在 0 到 999 999 999 999 之间的数字, 缺省值为 0。如果指定范围在 1 字节到 32 KB 的 SSL/TLS 密钥重置计数, 那么 SSL/TLS 通道将使用 32 KB 的密钥重置计数。这是为了避免小型 SSL/TLS 密钥重置值将发生的过多密钥重置的处理成本。

当发起通道 MCA 发送和接收的未加密字节总数超过指定值时, 或者如果在发送或接收数据之前启用了通道脉动信号 (以先发生者为准), 那么将重新协商密钥。

为重新协商而发送和接收的字节计数包括通道 MCA 发送和接收的控制信息, 每当发生重新协商时将重置这些信息。

使用值 0 指示永不重新协商密钥。

要确定此属性的值, 请将 MQIA_SSL_RESET_COUNT 选择器与 MQINQ 调用配合使用。

StartStop 事件 (MQLONG)

这将控制是否生成启动和停止事件。该值为下列其中之一:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息, 请参阅 [事件监视](#)。

要确定此属性的值, 请将 MQIA_START_STOP_EVENT 选择器与 MQINQ 调用配合使用。

StatisticsInterval (MQLONG)

这指定将统计信息监视数据写入监视队列的频率 (以秒计)。

该值是 0 到 604800 范围内的整数, 缺省值为 1800 (30 分钟)。

要确定此属性的值, 请将 MQIA_STATISTICS_INTERVAL 选择器与 MQINQ 调用配合使用。

SyncPoint (MQLONG)

这指示本地队列管理器是否支持使用 MQGET, MQPUT 和 MQPUT1 调用的工作单元和同步点。

MQSP_AVAILABLE

可用的工作单元和同步。

MQSP_NOT_AVAILABLE

工作单元和同步指向不可用。

- 在 z/OS 上，从不返回此值。

要确定此属性的值，请将 MQIA_SYNCPOINT 选择器与 MQINQ 调用配合使用。

TCPChannels (MQLONG)

这是使用 TCP/IP 传输协议的当前通道或可连接的客户端的最大通道数。

该值必须在范围 0 到 9999 之间，缺省值为 200。如果指定 0，那么不会使用 TCP/IP。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_TCP_CHANNELS 选择器与 MQINQ 调用配合使用。

TCPKeepAlive (MQLONG)

这指定是否使用 TCP KEEPALIVE 来检查连接的另一端是否仍然可用。如果它不可用，将关闭此通道。

该值为下列其中之一：

MQTCPKEEP_YES

使用 TCP 概要文件配置数据集中指定的 TCP KEEPALIVE。如果指定通道属性 KeepAlive 时间间隔 (KAINT)，那么将使用将其设置为的值。

MQTCPKEEP_NO

请勿使用 TCP KEEPALIVE。这是缺省值。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_TCP_KEEP_ALIVE 选择器与 MQINQ 调用配合使用。

TCPName (MQCHAR8)

这是您正在使用的唯一或缺省 TCP/IP 系统的名称，具体取决于 TCPStackType 的值。缺省值为 TCPIP。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQCA_TCP_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_TCP_NAME_LENGTH 指定。

TCPStackType (MQLONG)

这指定通道启动程序是只能使用 TCPName 中指定的 TCP/IP 地址空间，还是可以选择绑定到任何选定的 TCP/IP 地址

该值为下列其中之一：

MQTCPSTACK_SINGLE

通道启动程序只能使用 TCPName 中指定的 TCP/IP 地址空间。这是缺省值。

MQTCPSTACK_MULTIPLE

通道启动程序可以使用它可用的任何 TCP/IP 地址空间。如果没有为通道或侦听器指定其他任何值，那么缺省为 TCPName 中指定的值。

此属性仅在 z/OS 上受支持。

要确定此属性的值，请将 MQIA_TCP_STACK_TYPE 选择器与 MQINQ 调用配合使用。

TraceRoute 记录 (MQLONG)

这控制跟踪路由信息的记录。

该值为下列其中之一：

MQRECORDING_DISABLED

不允许追加到跟踪路由消息。

MQRECORDING_Q

将跟踪路由消息放入固定的指定队列。

MQRECORDING_MSG

将跟踪路由消息放入使用消息本身确定的队列。这是缺省值

要确定此属性的值，请将 MQIA_TRACE_ROUTE_录制选择器与 MQINQ 调用配合使用。

TriggerInterval (MQLONG)

这是用于限制触发器消息数的时间间隔 (以毫秒为单位)。仅当 *TriggerType* 为 MQTT_FIRST 时，这才相关。在这种情况下，通常仅当适当的消息到达队列时才会生成触发器消息，并且该队列先前为空。但是，在某些情况下，即使队列不为空，也可以使用 MQTT_FIRST 触发生成额外的触发器消息。生成这些附加触发器消息的频率不会超过每 *TriggerInterval* 毫秒。

有关触发的更多信息，请参阅 [触发通道](#)。

该值不小于 0 且不大于 999 999 999。缺省值为 999 999 999。

要确定此属性的值，请将 MQIA_TRIGGER_INTERVAL 选择器与 MQINQ 调用配合使用。

TriggerInterval (MQLONG)

这是用于限制触发器消息数的时间间隔 (以毫秒为单位)。仅当 *TriggerType* 为 MQTT_FIRST 时，这才相关。在这种情况下，通常仅当适当的消息到达队列时才会生成触发器消息，并且该队列先前为空。但是，在某些情况下，即使队列不为空，也可以使用 MQTT_FIRST 触发生成额外的触发器消息。生成这些附加触发器消息的频率不会超过每 *TriggerInterval* 毫秒。

有关触发的更多信息，请参阅 [触发通道](#)。

该值不小于 0 且不大于 999 999 999。缺省值为 999 999 999。

要确定此属性的值，请将 MQIA_TRIGGER_INTERVAL 选择器与 MQINQ 调用配合使用。

版本 (MQCFST)

这是作为 VVRRMMFF 的 WebSphere MQ 代码的版本，其中：

VV-版本

RR-发行版

MM-维护级别

FF-修订级别

XrCapability(MQLONG)

这将控制队列管理器是否支持 WebSphere MQ Telemetry 命令。

该值为下列其中之一：

MQCAP_SUPPORTED

支持已安装的 WebSphere MQ Telemetry 组件和 Telemetry 命令。

MQCAP_NOT_SUPPORTED

webSphere MQ Telemetry 组件未安装。

此属性仅在 IBM i, Unix 系统和 Windows 上受支持。

要确定此属性的值，请将 MQIA_XR_CAPABILITY 选择器与 MQINQ 调用配合使用。

队列的属性

有五种类型的队列定义。某些队列属性适用于所有类型的队列；其他队列属性仅适用于特定类型的队列。

队列类型

队列管理器支持以下类型的队列定义:

本地队列

您可以将消息存储在本地队列上。在 z/OS 上, 可以将其设置为共享或专用队列。

如果队列归程序连接到的队列管理器所有, 那么该队列对程序而言是本地队列。您可以从本地队列中获取消息及将消息放在本地队列上。

队列定义对象保存队列的定义信息以及放在该队列上的物理消息。

本地队列管理器队列

该队列存在于本地队列管理器上。该队列在 z/OS 上称为专用队列。

共享队列 (仅限 z/OS)

该队列存在于共享存储库中, 属于拥有该共享存储库的队列共享组的所有队列管理器都可访问该存储库。

连接到队列共享组中任何队列管理器的应用程序都可以在此类型的队列上放置消息并从队列中除去消息。此类队列实际上与本地队列相同。 *QType* 队列属性的值为 MQQT_LOCAL。

连接到本地队列管理器的应用程序可以将消息放在此类型的队列上并从队列中除去消息。 *QType* 队列属性的值为 MQQT_LOCAL。

集群队列

您可以将消息存储在定义该消息的队列管理器上的集群队列上。集群队列是由集群队列管理器托管并可供集群中其他队列管理器使用的队列。 *QType* 队列属性的值为 MQQT_CLUSTER。

集群队列定义将播发给集群中的其他队列管理器。集群中的其他队列管理器无需相应的远程队列定义即可将消息放入集群队列。可以使用集群名称列表在多个集群中播发集群队列。

在播发队列时, 集群中的任何队列管理器都可以将消息放入该队列中。要放入消息, 队列管理器必须从完整存储库中查明托管该队列的位置。然后, 它会将一些路由信息添加到消息, 并将消息放到集群传输队列上。

除了在 z/OS 上, 队列管理器可以将集群中其他队列管理器的消息存储在多个传输队列上。您可以采用两种不同的方式配置队列管理器以将消息存储在多个集群传输队列上。如果将队列管理器属性 DEFCLXQ 设置为 CHANNEL, 那么对于每个集群发送方通道, 将通过 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 自动创建一个不同的集群传输队列。如果将 CLCHNAME 传输队列选项设置为与一个或多个集群发送方通道匹配, 队列管理器可以将匹配通道的消息存储在该传输队列上。

集群队列可以是 IBM WebSphere MQ for z/OS 中队列共享组的成员所共享的队列。

远程队列

远程队列不是物理队列; 它是远程队列管理器上存在的队列的本地定义。远程队列的本地定义包含一些信息, 这些信息告诉本地队列管理器如何将消息路由到远程队列管理器。

连接到本地队列管理器的应用程序可以将消息放置在此类型的队列上; 这些消息放置在用于将消息路由到远程队列管理器的本地传输队列上。应用程序无法从远程队列中除去消息。 *QType* 队列属性的值为 MQQT_REMOTE。

您还可以将远程队列定义用于:

- 应答队列别名判别

在这种情况下, 定义的名称是应答队列的名称。有关更多信息, 请参阅 [应答队列别名和集群](#)。

- 队列管理器别名判别

在这种情况下, 定义的名称是队列管理器的别名, 而不是队列的名称。有关更多信息, 请参阅 [队列管理器别名和集群](#)。

别名队列

这不是物理队列; 它是本地队列, 共享队列, 集群队列或远程队列的备用名称。别名解析到的队列的名称是别名队列定义的一部分。

连接到本地队列管理器的应用程序可以将消息放置在此类型的队列上; 这些消息放置在别名解析到的队列上。如果别名解析为本地队列, 共享队列或具有本地实例的集群队列, 那么应用程序可以从此类型的队列中除去消息。QType 队列属性的值为 MQQT_ALIAS。

模型队列

这不是物理队列; 它是可从中创建本地队列的一组队列属性。

不能将消息存储在此类型的队列上。

队列属性

某些队列属性适用于所有类型的队列; 其他队列属性仅适用于特定类型的队列。属性应用于的队列类型显示在第 733 页的表 573 和后续表中。

第 733 页的表 573 汇总了特定于队列的属性。这些属性按字母顺序进行描述。

注: 本节中显示的属性名称是用于 MQINQ 和 MQSET 调用的描述性名称; 这些名称与 PCF 命令的名称相同。当 MQSC 命令用于定义, 改变或显示属性时, 将使用备用短名称; 请参阅 [脚本 \(MQSC\) 命令](#) 以获取详细信息。

表 573: 队列的属性. 这些列按如下所示应用:

- 本地队列的列也适用于共享队列。
- 模型队列的列指示从模型队列创建的本地队列继承哪些属性。
- 集群队列的列指示在打开集群队列以单独进行查询或进行查询和输出时可查询的属性。如果查询任何其他属性, 那么调用将返回完成代码 MQCC_WARNING 和原因码 MQRC_SELECTOR_NOT_FOR_TYPE (2068)。
如果打开集群队列以进行查询以及一个或多个输入, 浏览或设置, 那么将改为应用本地队列的列。
如果为单独查询或为查询和输出打开集群队列, 并指定基本队列管理器名称, 那么将改为应用本地队列的列。

| 属性 | 描述 | 本地 | 模型 | 别名 | 远程 | 集群 |
|--|-----------------------|----|----|----|----|----|
| AlterationDate | 上次更改定义的日期 | ✓ | | ✓ | ✓ | |
| AlterationTime | 上次更改定义的时间 | ✓ | | ✓ | ✓ | |
| BackoutRequeueQName | 过多的回退重新排队队列名称 | ✓ | ✓ | | | |
| BackoutThreshold | 回退阈值 | ✓ | ✓ | | | |
| BaseQName | 别名解析为的队列名称 | | | ✓ | | |
| CFStrucName | 耦合设施结构名称 | ✓ | ✓ | | | |
| CLCHNAME | 集群发送方通道名称 | ✓ | ✓ | | | |
| ClusterName | 队列所属的集群的名称 | ✓ | | ✓ | ✓ | ✓ |
| ClusterNameList | 包含队列所属集群的名称的名称列表对象的名称 | ✓ | | ✓ | ✓ | |
| CLWLQueuePriority | 集群工作负载队列优先级 | ✓ | | ✓ | ✓ | ✓ |
| CLWLQueueRank | 集群工作负载队列列组 | ✓ | | ✓ | ✓ | ✓ |
| CLWLUseQ | 使用远程队列 | ✓ | | | | |
| CreationDate | 创建队列的日期 | ✓ | | | | |
| CreationTime | 创建队列的时间 | ✓ | | | | |
| CurrentQDepth | 当前队列深度 | ✓ | | | | |
| DefaultPutResponse | 缺省放置响应 | ✓ | ✓ | ✓ | ✓ | |
| DefBind | 缺省绑定 | ✓ | | ✓ | ✓ | ✓ |
| DefinitionType attribute | 队列定义类型 | ✓ | ✓ | | | |
| DefInputOpenOption | 缺省输入打开选项 | ✓ | ✓ | | | |
| DefPersistence | 缺省消息持久性 | ✓ | ✓ | ✓ | ✓ | ✓ |

表 573: 队列的属性. 这些列按如下所示应用:

- 本地队列的列也适用于共享队列。
- 模型队列的列指示从模型队列创建的本地队列继承哪些属性。
- 集群队列的列指示在打开集群队列以单独进行查询或进行查询和输出时可查询的属性。如果查询任何其他属性，那么调用将返回完成代码 MQCC_WARNING 和原因码 MQRC_SELECTOR_NOT_FOR_TYPE (2068)。

如果打开集群队列以进行查询以及一个或多个输入，浏览或设置，那么将改为应用本地队列的列。

如果为单独查询或为查询和输出打开集群队列，并指定基本队列管理器名称，那么将改为应用本地队列的列。

(继续)

| 属性 | 描述 | 本地 | 模型 | 别名 | 远程 | 集群 |
|---|--------------------------------|----|----|----|----|----|
| DefPriority | 缺省消息优先级 | ✓ | ✓ | ✓ | ✓ | ✓ |
| DefReadAhead | 缺省提前读取 | ✓ | ✓ | ✓ | | |
| DistLists | 分发列表支持 | ✓ | ✓ | | | |
| HardenGetBackout | 是否保持准确的回退计数 | ✓ | ✓ | | | |
| IndexType | 索引类型 | ✓ | ✓ | | | |
| InhibitGet | 是否允许对队列执行 get 操作 | ✓ | ✓ | ✓ | | |
| InhibitPut | 是否允许队列的放置操作 | ✓ | ✓ | ✓ | ✓ | ✓ |
| InitiationQName | 启动队列的名称 | ✓ | ✓ | | | |
| MaxMsgLength | 最大消息长度 (字节) | ✓ | ✓ | | | |
| MaxQDepth | 最大队列深度 | ✓ | ✓ | | | |
| MsgDeliverySequence attribute | 消息传递顺序 | ✓ | ✓ | | | |
| NonPersistentMessage Class | 非持久消息的可靠性目标 | ✓ | ✓ | | | |
| OpenInputCount | 输入的打开次数 | ✓ | | | | |
| OpenOutputCount | 为输出打开的次数 | ✓ | | | | |
| PropertyControl | 属性控制 | ✓ | ✓ | ✓ | | |
| ProcessName | 进程名称 | ✓ | ✓ | | | |
| QDepthHighEvent attribute | 是否生成 "队列深度高" 事件 | ✓ | ✓ | | | |
| QDepthHighLimit | 队列深度的上限 | ✓ | ✓ | | | |
| QDepthLowEvent attribute | 是否生成队列深度下限事件 | ✓ | ✓ | | | |
| QDepthLowLimit attribute | 队列深度的下限 | ✓ | ✓ | | | |
| QDepthMaxEvent | 是否生成队列已满事件 | ✓ | ✓ | | | |
| QDesc | 队列描述 | ✓ | ✓ | ✓ | ✓ | ✓ |
| QName | 队列名称 | ✓ | | ✓ | ✓ | ✓ |
| QServiceInterval | 队列服务时间间隔的目标 | ✓ | ✓ | | | |
| QServiceIntervalEvent attribute | 是生成 "服务时间间隔高" 还是 "服务时间间隔正常" 事件 | ✓ | ✓ | | | |
| QSGDisp attribute | 队列共享组处置 | ✓ | | ✓ | ✓ | |
| QueueAccounting | 队列记帐数据收集 | ✓ | ✓ | ✓ | ✓ | ✓ |
| QueueMonitoring | 队列的联机监视数据 | ✓ | ✓ | | | |
| QueueStatistics | 队列统计信息数据收集 | ✓ | ✓ | ✓ | ✓ | ✓ |
| QType | 队列类型 | ✓ | | ✓ | ✓ | ✓ |
| RemoteQMgrName | 远程队列管理器的名称 | | | | ✓ | |

表 573: 队列的属性. 这些列按如下所示应用:

- 本地队列的列也适用于共享队列。
 - 模型队列的列指示从模型队列创建的本地队列继承哪些属性。
 - 集群队列的列指示在打开集群队列以单独进行查询或进行查询和输出时可查询的属性。如果查询任何其他属性，那么调用将返回完成代码 MQCC_WARNING 和原因码 MQRC_SELECTOR_NOT_FOR_TYPE (2068)。
- 如果打开集群队列以进行查询以及一个或多个输入，浏览或设置，那么将改为应用本地队列的列。
- 如果为单独查询或为查询和输出打开集群队列，并指定基本队列管理器名称，那么将改为应用本地队列的列。

(继续)

| 属性 | 描述 | 本地 | 模型 | 别名 | 远程 | 集群 |
|------------------------------------|------------------|----|----|----|----|----|
| RemoteQName | 远程队列的名称 | | | | ✓ | |
| RetentionInterval | 保留时间间隔 | ✓ | ✓ | | | |
| Scope | 队列的条目是否也存在于单元目录中 | ✓ | | ✓ | ✓ | |
| Shareability | 队列可共享性 | ✓ | ✓ | | | |
| StorageClass | 队列的存储类 | ✓ | ✓ | | | |
| TriggerControl | 触发器控制 | ✓ | ✓ | | | |
| TriggerData | 触发器数据 | ✓ | ✓ | | | |
| TriggerDepth | 触发器深度 | ✓ | ✓ | | | |
| TriggerMsgPriority | 触发器的阈值消息优先级 | ✓ | ✓ | | | |
| TriggerType | 触发器类型 | ✓ | ✓ | | | |
| Usage attribute | 队列使用情况 | ✓ | ✓ | | | |
| XmitQName | 传输队列的名称 | | | | ✓ | |

相关概念

[集群队列](#)

[本地队列](#)

AlterationDate (MQCHAR12)

上次更改定义日期。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | |

这是上次更改定义日期。日期的格式为 YYYY-MM-DD，用两个尾部空格填充，使长度为 12 个字节 (例如，1992-09-23--，其中 -- 表示两个空白字符)。

当队列管理器运行时，某些属性 (例如，*CurrentQDepth*) 的值会更改。对这些属性的更改不会影响 *AlterationDate*。

要确定此属性的值，请将 MQCA_ALTERATION_DATE 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_DATE_LENGTH 提供。

AlterationTime (MQCHAR8)

上次更改定义的时间。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | |

这是上次更改定义的时间。时间的格式为 HH.MM.SS (使用 24 小时制时钟)，如果小时小于 10 (例如 09.10.20)，那么为前导零。

- 在 z/OS 上，时间为格林威治标准时间 (GMT)，这取决于系统时钟是否准确设置为 GMT。
- 在其他环境中，时间是本地时间。

当队列管理器运行时，某些属性 (例如，*CurrentQDepth*) 的值会更改。对这些属性的更改不会影响 *AlterationTime*。

要确定此属性的值，请将 MQCA_ALTERATION_TIME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_TIME_LENGTH 提供。

BackoutRequeueQName (MQCHAR48)

这是过多的回退重排队列名称。除了允许查询其值外，队列管理器不会根据此属性的值执行任何操作。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

在 WebSphere Application Server 中运行的应用程序以及使用 WebSphere MQ Application Server 工具的应用程序使用此属性来确定回退消息的位置。对于所有其他应用程序，队列管理器不会根据属性的值执行任何操作。

WebSphere MQ classes for JMS 使用此属性来确定将已回退的消息传输到 *BackoutThreshold* 属性所指定的最大次数的位置。

要确定此属性的值，请将 MQCA_BACKOUT_REQ_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

BackoutThreshold (MQLONG)

这是回退阈值。除了允许查询其值外，队列管理器不会根据此属性的值执行任何操作。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

在 WebSphere Application Server 中运行的应用程序以及使用 WebSphere MQ Application Server 工具的应用程序将使用此属性来确定是否应该回退消息。对于所有其他应用程序，队列管理器不会根据属性的值执行任何操作。

WebSphere MQ classes for JMS 使用此属性来确定在将消息传输到 *BackoutRequeueQName* 属性指定的队列之前允许回退消息的次数。

要确定此属性的值，请将 MQIA_BACKOUT_THRESHOLD 选择器与 MQINQ 调用配合使用。

BaseQName (MQCHAR48)

这是对本地队列管理器定义的队列的名称。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| | | X | | |

(有关队列名称的更多信息，请参阅 [MQOD- ObjectName](#) 字段。) 队列是下列其中一种类型：

MQQT_LOCAL

本地队列。

MQQT_REMOTE

远程队列的本地定义。

MQQT_CLUSTER

集群队列。

要确定此属性的值，请将 MQCA_BASE_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

BaseType (MQCFIN)

别名解析为的对象类型。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| | | X | | |

它是下列其中一个值:

MQOT_Q

基本对象类型是队列

MQOT_TOPIC

基本对象类型是主题

CFStrucName (MQCHAR12)

这是用于存储队列上的消息的耦合设施结构的名称。名称的第一个字符在 A 到 Z 范围内，其余字符在 A 到 Z 范围内，0 到 9 或空白。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

要获取耦合设施中结构的全名，请使用 CFStrucName 队列属性的值作为 QSGName 队列管理器属性的值的后缀。

此属性仅适用于共享队列; 如果 QSGDisp 没有值 MQQSGD_SHARED，那么将忽略此属性。

要确定此属性的值，请将 MQCA_CF_STRUC_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_CF_STRUC_NAME_LENGTH 指定。

此属性仅在 z/OS 上受支持。

ClusterChannel 名称 (MQCHAR20)

ClusterChannelName 是将此队列用作传输队列的集群发送方通道的通用名称。该属性指定哪些集群发送方通道将消息从此集群传输队列发送到集群接收方通道。ClusterChannel 名称在 z/OS 上不受支持。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| ✓ | ✓ | | | |

缺省队列管理器配置是，所有集群发送方通道从单个传输队列 SYSTEM.CLUSTER.TRANSMIT.QUEUE 发送消息。可以通过更改队列管理器属性 DefClusterXmitQueueType 来更改缺省配置。该属性的缺省值为 SCTQ。您可以将值更改为 CHANNEL。如果将 DefClusterXmitQueueType 属性设置为 CHANNEL，那么每个集群发送方通道缺省为使用特定的集群传输队列 SYSTEM.CLUSTER.TRANSMIT.ChannelName。

您还可以手动将传输队列属性 ClusterChannelName 设置为集群发送方通道。发往通过集群发送方通道连接的队列管理器的消息将存储在识别集群发送方通道的传输队列中，而不会存储在缺省集群传输队列中。如果将 ClusterChannelName 属性设置为空白，那么通道在重新启动时将切换至缺省集群传输队列。缺省队列为 SYSTEM.CLUSTER.TRANSMIT.ChannelName 或 SYSTEM.CLUSTER.TRANSMIT.QUEUE，这取决于队列管理器 DefClusterXmitQueueType 属性的值。

通过在 ClusterChannelName 中指定星号“*”，可以将传输队列与一组集群发送方通道相关联。星号可以位于通道名称字符串的开头、结尾或中间任意位置。ClusterChannelName 的长度限于 20 个字符: MQ_CHANNEL_NAME_LENGTH。

ClusterName (MQCHAR48)

这是队列所属的集群的名称。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | X |

如果队列属于多个集群，那么 *ClusterNameList* 指定用于标识集群的名称列表对象的名称，而 *ClusterName* 为空白。*ClusterName* 和 *ClusterNameList* 中的至少一个必须为空白。

要确定此属性的值，请将 MQCA_CLUSTER_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_CLUSTER_NAME_LENGTH 指定。

ClusterNameList (MQCHAR48)

这是包含此队列所属集群的名称的名称列表对象的名称。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | |

如果队列仅属于一个集群，那么名称列表对象仅包含一个名称。或者，可以使用 *ClusterName* 来指定集群的名称，在这种情况下，*ClusterNameList* 为空白。*ClusterName* 和 *ClusterNameList* 中的至少一个必须为空白。

要确定此属性的值，请将 MQCA_CLUSTER_NAMELIST 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_NAMELIST_NAME_LENGTH 提供。

CLWLQueuePriority (MQLONG)

这是集群工作负载队列优先级，范围在 0 到 9 之间的值表示队列的优先级。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | X |

有关更多信息，请参阅 [集群队列](#)。

要确定此属性的值，请将 MQIA_CLWL_Q_PRIORITY 选择器与 MQINQ 调用配合使用。

CLWLQueueRank (MQLONG)

这是集群工作负载队列列组，范围在 0 到 9 之间的值表示队列的列组。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | X |

有关更多信息，请参阅 [集群队列](#)。

要确定此属性的值，请将 MQIA_CLWL_Q_RANK 选择器与 MQINQ 调用配合使用。

CLWLUseQ (MQLONG)

这定义了当目标队列同时具有本地实例和至少一个远程集群实例时 MQPUT 的行为。如果放置操作起源于集群通道，那么此属性不适用。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | | | |

该值为下列其中之一：

MQCLWL_USEQ_ANY

使用远程队列和本地队列。

MQCLWL_USEQ_LOCAL

请勿使用远程队列。

MQCLWL_USEQ_AS_Q_MGR

从队列管理器的 MQIA_CLWL_USEQ 继承定义。

有关更多信息，请参阅 [集群队列](#)。

要确定此属性的值，请将 MQCA_CLWL_USEQ 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_CLWL_USEQ_LENGTH 提供。

CreationDate (MQCHAR12)

这是创建队列的日期。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | | | |

日期的格式为 YYYY-MM-DD，用两个尾部空格填充，使长度为 12 个字节 (例如，2013-09-23，其中 表示 2 个空白字符)。

- 在 IBM i 上，队列的创建日期可能与表示该队列的底层操作系统实体 (文件或用户空间) 的创建日期不同。

要确定此属性的值，请将 MQCA_CREATION_DATE 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_CREATION_DATE_LENGTH 提供。

CreationTime (MQCHAR8)

这是创建队列的时间。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | | | |

时间的格式为 HH.MM.SS (使用 24 小时制时钟)，如果小时小于 10 (例如 09.10.20)，那么为前导零。

- 在 z/OS 上，时间为格林威治标准时间 (GMT)，这取决于系统时钟是否准确设置为 GMT。
- 在其他环境中，时间是本地时间。
- 在 IBM i 上，队列的创建时间可能与表示该队列的底层操作系统实体 (文件或用户空间) 的创建时间不同。

要确定此属性的值，请将 MQCA_CREATION_TIME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_CREATION_TIME_LENGTH 提供。

CurrentQDepth (MQLONG)

这是当前在队列上的消息数。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | | | |

它在 MQPUT 调用期间和 MQGET 调用回退期间递增。在非浏览 MQGET 调用期间以及在回退 MQPUT 调用期间，将对其进行递减。这样做的效果是，计数包括已放在工作单元中的队列上但尚未落实的消息，即使这些消息没有资格由 MQGET 调用检索。同样，它会排除使用 MQGET 调用在工作单元中检索到但尚未落实的消息。

此计数还包括已超过到期时间但尚未废弃的消息，尽管这些消息不适合检索。请参阅 [MQMD-到期字段](#) 以获取更多信息。

工作单元处理和消息分段都可能导致 *CurrentQDepth* 超过 *MaxQDepth*。但是，这不会影响消息的可检索性；可以通过正常方式使用 MQGET 调用来检索队列上的所有消息。

此属性的值在队列管理器运行时波动。

要确定此属性的值，请将 MQIA_CURRENT_Q_DEPTH 选择器与 MQINQ 调用配合使用。

DefaultPut 响应 (MQLONG)

指定当应用程序指定 MQPMO_RESPONSE_AS_Q_DEF 时要用于将操作放入队列的响应类型。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | X | |

它是下列其中一个值:

MQPRT_SYNC_RESPONSE

同步发出 put 操作, 返回响应。

MQPRT_ASYNC_RESPONSE

异步发出 put 操作, 返回 MQMD 字段的子集。

DefBind (MQLONG)

这是在 MQOPEN 调用上指定 MQOO_BIND_AS_Q_DEF 并且队列是集群队列时使用的缺省绑定。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | X |

该值为下列其中之一:

MQBND_BIND_ON_OPEN

由 MQOPEN 调用修订的绑定。

MQBND_BIND_NOT_FIXED

绑定未固定。

MQBND_BIND_ON_GROUP

允许应用程序请求将一组消息全部分配给同一目标实例。由于此值是 IBM WebSphere MQ Version 7.1 中的新增值, 因此如果打开此队列的任何应用程序正在连接到 IBM WebSphere MQ Version 7.0.1 或更低版本的队列管理器, 那么不得使用此值。

要确定此属性的值, 请将 MQIA_DEF_BIND 选择器与 MQINQ 调用配合使用。

DefinitionType (MQLONG)

这指示队列的定义方式。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

该值为下列其中之一:

MQQDT_PREDEFINED

该队列是由系统管理员创建的永久队列; 只有系统管理员才能将其删除。

预定义队列是使用 DEFINE MQSC 命令创建的, 只能使用 DELETE MQSC 命令删除。无法从模型队列创建预定义队列。

命令可以由操作员发出, 也可以由向命令输入队列发送命令消息的授权用户发出 (请参阅 [CommandInputQName 属性](#) 以获取更多信息)。

MQQDT_PERMANENT_DYNAMIC

该队列是由发出 MQOPEN 调用的应用程序创建的永久队列, 该调用具有对象描述符 MQOD 中指定的模型队列的名称。模型队列定义具有 *DefinitionType* 属性的值 MQQDT_PERMANENT_DYNAMIC。

可以使用 MQCLOSE 调用来删除此类型的队列。请参阅第 568 页的『MQCLOSE-关闭对象』, 以获取更多详细信息。

永久动态队列的 *QSGDisp* 属性值为 MQQSGD_Q_MGR。

MQQDT_TEMPORARY_DYNAMIC

该队列是由发出 MQOPEN 调用的应用程序创建的临时队列，该调用具有对象描述符 MQOD 中指定的模型队列的名称。模型队列定义具有 *DefinitionType* 属性的值 MQQDT_TEMPORARY_DYNAMIC。

此类型的队列由 MQCLOSE 调用在由创建该队列的应用程序关闭时自动删除。

临时动态队列的 *QSGDisp* 属性值为 MQQSGD_Q_MGR。

MQQDT_SHARED_DYNAMIC

该队列是由发出 MQOPEN 调用的应用程序创建的共享永久队列，该调用具有对象描述符 MQOD 中指定的模型队列的名称。模型队列定义具有 *DefinitionType* 属性的值 MQQDT_SHARED_DYNAMIC。

可以使用 MQCLOSE 调用来删除此类型的队列。请参阅第 568 页的『MQCLOSE-关闭对象』，以获取更多详细信息。

共享动态队列的 *QSGDisp* 属性值为 MQQSGD_SHARED。

模型队列定义中的此属性不指示模型队列的定义方式，因为模型队列始终是预定义的。相反，模型队列中此属性的值用于确定使用 MQOPEN 调用从模型队列定义创建的每个动态队列的 *DefinitionType*。

要确定此属性的值，请将 MQIA_DEFINITION_TYPE 选择器与 MQINQ 调用配合使用。

DefInputOpenOption (MQLONG)

这是打开队列以进行输入的缺省方式。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

如果在打开队列时在 MQOPEN 调用上指定了 MQOO_INPUT_AS_Q_DEF 选项，那么它适用。该值为下列其中之一：

MQOO_INPUT_EXCLUSIVE

打开队列以获取具有独占访问权的消息。

打开队列以与后续 MQGET 调用配合使用。如果此应用程序或其他应用程序当前打开队列以用于任何类型的输入 (MQOO_INPUT_SHARED 或 MQOO_INPUT_EXCLUSIVE)，那么调用将失败，原因码为 MQRC_OBJECT_IN_USE。

MQOO_INPUT_SHARED

打开队列以获取具有共享访问权的消息。

打开队列以与后续 MQGET 调用配合使用。如果队列当前由此应用程序或另一个使用 MQOO_INPUT_SHARED 的应用程序打开，但如果队列当前使用 MQOO_INPUT_EXCLUSIVE 打开，那么调用可能成功失败，原因码为 MQRC_OBJECT_IN_USE。

要确定此属性的值，请将 MQIA_DEF_INPUT_OPEN_OPTION 选择器与 MQINQ 调用配合使用。

DefPersistence (MQLONG)

这是队列中消息的缺省持久性。如果在放入消息时在消息描述符中指定了 MQPER_PERSISTENCE_AS_Q_DEF，那么它适用。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | X | X |

如果队列名称解析路径中有多个定义，那么缺省持久性取自 MQPUT 或 MQPUT1 调用时路径中第一个定义中此属性的值。它可以是：

- 别名队列
- 本地队列
- 远程队列的本地定义
- 队列管理器别名

- 传输队列 (例如, *DefXmitQName* 队列)

该值为下列其中之一:

MQPER_PERSISTENT

消息在系统故障和队列管理器重新启动后仍然存在。无法将持久消息放在以下位置:

- 临时动态队列数
- 映射到 CFLEVEL (\$TAG1) 或更低级别的 CFSTRUCT 对象的共享队列, 或者 CFSTRUCT 对象定义为 RECOVER (NO) 的共享队列。

持久消息可以放置在永久动态队列和预定义队列上。

MQPER_NOT_PERSISTENT

该消息通常不会在系统故障或队列管理器重新启动后继续存在。即使在队列管理器重新启动期间在辅助存储器上找到消息的完整副本, 这也适用。

对于共享队列, 非持久消息 *do* 会在队列共享组中的队列管理器重新启动后存活下来, 但不会在用于在共享队列上存储消息的耦合设施发生故障后存活下来。

持久消息和非持久消息都可以存在于同一队列中。

要确定此属性的值, 请将 MQIA_DEF_PERSISTENCE 选择器与 MQINQ 调用配合使用。

DefPriority (MQLONG)

这是队列上消息的缺省优先级。如果将消息放入队列时在消息描述符中指定了 MQPRI_PRIORITY_AS_Q_DEF, 那么这适用。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | X | X |

如果在队列名解析路径中有多个定义, 那么在执行 put 操作时, 将从路径中的 *first* 定义中的此属性值获取消息的缺省优先级。它可以是:

- 别名队列
- 本地队列
- 远程队列的本地定义
- 队列管理器别名
- 传输队列 (例如, *DefXmitQName* 队列)

将消息放入队列的方式取决于队列的 *MsgDeliverySequence* 属性的值:

- 如果 *MsgDeliverySequence* 属性为 MQMDS_PRIORITY, 那么将消息放入队列的逻辑位置取决于消息描述符中 *Priority* 字段的值。
- 如果 *MsgDeliverySequence* 属性是 MQMDS_FIFO, 那么会将消息放在队列上, 就好像它们的优先级等于已解析队列的 *DefPriority* 一样, 而不考虑消息描述符中 *Priority* 字段的值。但是, *Priority* 字段保留由放置消息的应用程序指定的值。请参阅 [MsgDeliverySequence 属性](#) 以获取更多信息。

优先级在范围零 (最低) 到 *MaxPriority* (最高) 之间; 请参阅 [MaxPriority 属性](#)。

要确定此属性的值, 请将 MQIA_DEF_PRIORITY 选择器与 MQINQ 调用配合使用。

DefReadAhead (MQLONG)

指定传递到客户机的非持久消息的缺省预读行为。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | | |

DefReadAhead 可以设置为下列其中一个值:

MQREADA_NO

在应用程序请求非持久消息之前，不会将这些消息提前发送到客户机。如果客户机异常中止，最多只会丢失一个非持久消息。

MQREADA_YES

在应用程序请求非持久消息之前，会将这些消息提前发送到客户机。如果客户机异常结束，或者如果客户机未使用其发送的所有消息，那么可能会丢失非持久消息。

MQREADA_DISABLED

未对此队列启用非持久消息预读。无论客户机应用程序是否请求预读，都不会将消息发送到客户机。

要确定此属性的值，请将 MQIA_DEF_READ_AHEAD 选择器与 MQINQ 调用配合使用。

DefPResp (MQLONG)

当 MQPMO 中的 PutResponse 类型设置为 MQPMO_RESPONSE_AS_Q_DEF 时，缺省放置响应类型 (DEFpresP) 属性定义应用程序使用的值。此属性对所有队列类型都有效。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|-----|-----|-----|-----|-----|
| Yes | Yes | Yes | Yes | Yes |

该值为下列其中之一：

同步

将同步发出 put 操作以返回响应。

异步

异步发出 put 操作，返回 MQMD 字段的子集。

要确定此属性的值，请将 MQIA_DEF_PUT_RESPONSE_TYPE 选择器与 MQINQ 调用配合使用。

DistLists (MQLONG)

这指示是否可以将分发列表消息放在队列上。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

消息通道代理程序 (MCA) 设置此属性以通知本地队列管理器通道另一端的队列管理器是否支持分发列表。后一个队列管理器 (称为 伙伴关系 队列管理器) 是发送 MCA 将消息从本地传输队列中除去后下一个接收消息的队列管理器。

每当发送 MCA 与伙伴队列管理器上的接收 MCA 建立连接时，发送 MCA 都会设置该属性。这样，发送 MCA 会导致本地队列管理器仅将伙伴队列管理器可正确处理的消息放入传输队列。

此属性主要用于与传输队列配合使用，但将执行所描述的处理，而不考虑为队列定义的用法 (请参阅 "用法" 属性)。

该值为下列其中之一：

MQDL_SUPPORTED

分发列表消息可以存储在队列上，并以该格式传输到伙伴队列管理器。这将减少将消息发送到多个目标所需的处理量。

MQDL_NOT_SUPPORTED

由于伙伴队列管理器不支持分发列表，因此无法将分发列表消息存储在队列上。如果应用程序放置分发列表消息，并且该消息将放置在此队列上，那么队列管理器将分割分发列表消息并将各个消息改为放置在队列上。这将增加将消息发送到多个目标所需的处理量，但确保伙伴队列管理器正确处理消息。

要确定此属性的值，请将 MQIA_DIST_LISTS 选择器与 MQINQ 调用配合使用。要更改此属性的值，请使用 MQSET 调用。

此属性在 z/OS 上不受支持。

HardenGet 回退 (MQLONG)

对于每条消息，将保留由工作单元中的 MQGET 调用检索消息的次数以及随后回退该工作单元的次数。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

完成 MQGET 调用后，此计数在消息描述符中的 *BackoutCount* 字段中可用。

消息回退计数在队列管理器重新启动后仍然存在。但是，为了确保计数准确，每次 MQGET 调用在此队列的工作单元中检索消息时，都必须固化信息 (记录在磁盘或其他永久存储设备上)。如果未执行此操作，队列管理器将失败，并且 MQGET 调用将回退，那么计数可能递增，也可能不递增。

但是，工作单元中每个 MQGET 调用的固化信息会施加额外的处理成本，因此仅当计数必须准确时，才会将 *HardenGetBackout* 属性设置为 MQQA_BACKOUT_HARDENED。

在 IBM i, UNIX 系统和 Windows 上，始终会固化消息回退计数，而不考虑此属性的设置。

可能的值如下所示：

MQQA_BACKOUT_HARDENED

固化用于确保此队列中消息的回退计数准确。

MQQA_BACKOUT_NOT_HARDENED

固化不用于确保此队列上消息的回退计数是准确的。因此，计数可能低于应该的计数。

要确定此属性的值，请将 MQIA_HARDEN_GET_BACKOUT 选择器与 MQINQ 调用配合使用。

IndexType (MQLONG)

这指定队列管理器为队列上的消息维护的索引类型。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

所需的索引类型取决于应用程序检索消息的方式，以及队列是共享队列还是非共享队列 (请参阅 *QSGDisp* 属性)。对于 *IndexType*，可以使用以下值：

MQIT_NONE

此队列的队列管理器未维护任何索引。将此值用于通常按顺序处理的队列，即，不使用 MQGET 调用上的任何选择标准。

MQIT_MSG_ID

队列管理器维护使用队列上消息的消息标识的索引。使用此值队列，其中应用程序通常使用消息标识作为 MQGET 调用上的选择条件来检索消息。

MQIT_CORREL_ID

队列管理器维护使用队列上消息的相关标识的索引。将此值用于应用程序通常使用相关标识作为 MQGET 调用的选择条件来检索消息的队列。

MQIT_MSG_TOKEN

队列管理器维护一个索引，该索引使用队列上消息的消息标记，以便与 z/OS 的工作负载管理器 (WLM) 功能配合使用。

必须为 WLM 管理的队列指定此选项；请勿为任何其他类型的队列指定此选项。此外，如果应用程序未使用 z/OS 工作负载管理器函数，但正在使用消息令牌作为 MQGET 调用的选择条件来检索消息，请不要将此值用于该队列。

MQIT_GROUP_ID

队列管理器维护使用队列上消息的组标识的索引。此值必须用于应用程序使用 MQGET 调用上的 MQGMO_LOGICAL_ORDER 选项检索消息的队列。

具有此索引类型的队列不能是传输队列。必须定义具有此索引类型的共享队列以映射到 CFLEVEL (\$TAG1) 或 CFLEVEL (4) 的 CFSTRUCT 对象。

注：

1. 未定义索引类型为 MQIT_GROUP_ID 的队列上消息的物理顺序，因为使用 MQGET 调用上的 MQGMO_LOGICAL_ORDER 选项对队列进行了优化以高效检索消息。这意味着消息的物理顺序通常不是消息到达队列的顺序。
2. If an MQIT_GROUP_ID queue has a *MsgDeliverySequence* of MQMDS_PRIORITY, the queue manager uses message priorities 0 and 1 to optimize the retrieval of messages in logical order. 因此，组中的第一条消息不得具有 0 或 1 的优先级；如果具有 0 或 1 的优先级，那么将处理该消息，就好像它具有 2 的优先级一样。MQMD 结构中的 *Priority* 字段未更改。

有关消息组的更多信息，请参阅 [MQGMO-选项](#) 字段中组和段选项的描述。

应在各种情况下使用的索引类型显示在 [第 745 页的表 574](#) 和 [第 746 页的表 575](#) 中。

| 表 574: 未指定 MQGMO_LOGICAL_ORDER 时队列索引类型的建议值或必需值 | | |
|--|---|---|
| MQGET 调用上的选择条件 | 非共享队列的索引类型 | 共享队列的索引类型 |
| None | 任何 | 任何 |
| 使用一个标识进行选择: | | |
| 消息标识 | MQIT_MSG_ID 建议 | 需要 MQIT_NONE 或 MQIT_MSG_ID; 建议使用 MQIT_MSG_ID |
| 相关标识 | MQIT_CORREL_ID 建议 | 需要 MQIT_CORREL_ID |
| 组标识 | MQIT_GROUP_ID 建议 | 需要 MQIT_GROUP_ID |
| 使用两个标识进行选择: | | |
| 消息标识加上相关标识 | MQIT_MSG_ID 或 MQIT_CORREL_ID 建议 | 需要 MQIT_NONE 或 MQIT_MSG_ID 或 MQIT_CORREL_ID (为了提高效率，建议选择索引类型以匹配将具有最不同键的 MQMD 字段) |
| 消息标识和组标识 | MQIT_MSG_ID 或 MQIT_GROUP_ID 建议 | 不支持 |
| 相关标识和组标识 | MQIT_CORREL_ID 或 MQIT_GROUP_ID 建议 | 不支持 |
| 使用三个标识进行选择: | | |
| 消息标识加相关标识加组标识 | MQIT_MSG_ID 或 MQIT_CORREL_ID 或 MQIT_GROUP_ID 建议 | 不支持 |
| 使用与组相关的条件进行选择: | | |
| 组标识和消息序号 | 需要 MQIT_GROUP_ID | 需要 MQIT_GROUP_ID |
| 消息序号 (必须为 1) | 需要 MQIT_GROUP_ID | 需要 MQIT_GROUP_ID |
| 使用消息令牌进行选择: | | |
| 应用程序使用的消息令牌 | 请勿使用 MQIT_MSG_TOKEN | |
| 用于 WLM 的消息令牌 | 需要 MQIT_MSG_TOKEN | 不支持 |

表 575: 指定 MQGMO_LOGICAL_ORDER 时队列索引类型的建议值或必需值

| MQGET 调用上的选择条件 | 非共享队列的索引类型 | 共享队列的索引类型 |
|--------------------|------------------|------------------|
| None | 需要 MQIT_GROUP_ID | 需要 MQIT_GROUP_ID |
| 使用一个标识进行选择: | | |
| 消息标识 | 需要 MQIT_GROUP_ID | 不支持 |
| 相关标识 | 需要 MQIT_GROUP_ID | 不支持 |
| 组标识 | 需要 MQIT_GROUP_ID | 需要 MQIT_GROUP_ID |
| 使用两个标识进行选择: | | |
| 消息标识加上相关标识 | 需要 MQIT_GROUP_ID | 不支持 |
| 消息标识和组标识 | 需要 MQIT_GROUP_ID | 不支持 |
| 相关标识和组标识 | 需要 MQIT_GROUP_ID | 不支持 |
| 使用三个标识进行选择: | | |
| 消息标识加相关标识加组标识 | 需要 MQIT_GROUP_ID | 不支持 |

要确定此属性的值，请将 MQIA_INDEX_TYPE 选择器与 MQINQ 调用配合使用。

此属性仅在 z/OS 上受支持。

InhibitGet (MQLONG)

这将控制是否允许对此队列执行 get 操作。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | | |

如果队列是别名队列，那么在执行 get 操作时，必须允许对别名和基本队列执行 get 操作，MQGET 调用才能成功。该值为下列其中之一：

MQQA_GET_INHIBITED

禁止获取操作。

MQGET 调用失败，原因码为 MQRC_GET_禁止。这包括指定 MQGMO_BROWSE_FIRST 或 MQGMO_BROWSE_NEXT 的 MQGET 调用。

注: 如果在工作单元中运行的 MQGET 调用成功完成，那么随后将 *InhibitGet* 属性的值更改为 MQQA_GET_所在值不会阻止落实该工作单元。

MQQA_GET_ALLOWED

允许执行获取操作。

要确定此属性的值，请将 MQIA_禁止获取选择器与 MQINQ 调用配合使用。要更改此属性的值，请使用 MQSET 调用。

InhibitPut (MQLONG)

这将控制是否允许对此队列执行放置操作。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | X | X |

如果队列名称解析路径中有多个定义，那么在执行 put 操作时，必须允许对路径中的每个定义 (包括任何队列管理器别名定义) 执行 put 操作，以便 MQPUT 或 MQPUT1 调用成功。该值为下列其中之一：

MQQA_PUT_INHIBITED

禁止执行放置操作。

MQPUT 和 MQPUT1 调用失败，原因码为 MQRC_PUT_禁止。

注: 如果在工作单元中运行的 MQPUT 调用成功完成，那么随后将 *InhibitPut* 属性的值更改为 MQQA_PUT_ALLOWED 不会阻止落实工作单元。

MQQA_PUT_ALLOWED

允许执行放置操作。

要确定此属性的值，请将 MQIA_禁止放置选择器与 MQINQ 调用配合使用。要更改此属性的值，请使用 MQSET 调用。

InitiationQName (MQCHAR48)

这是在本地队列管理器上定义的队列的名称; 该队列必须为 MQQT_LOCAL 类型。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | | | |

当由于消息到达此属性所属的队列而需要应用程序启动时，队列管理器将触发器消息发送到启动队列。启动队列必须由触发器监视器应用程序监视，该应用程序在收到触发器消息后启动相应的应用程序。

要确定此属性的值，请将 MQCA_INITIATION_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

MaxMsg 长度 (MQLONG)

这是可放置在队列上的最长物理消息的长度上限。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

但是，由于可以独立于 *MaxMsgLength* 队列管理器属性设置 *MaxMsgLength* 队列属性，因此可以放置在队列上的最长物理消息的长度的实际上限是这两个值中的较小者。

如果队列管理器支持分段，那么应用程序可以放置比两个 *MaxMsgLength* 属性中的较小属性长的逻辑消息，但仅当应用程序在 MQMD 中指定了 MQMF_SEGMENTATION_ALLOWED 标志时才可以这样做。如果指定了该标志，那么逻辑消息的长度上限为 999 999 999 字节，但通常由操作系统或运行应用程序的环境施加的资源约束会导致下限。

尝试将过长的消息放入队列失败，原因码如下之一:

- MQRC_MSG_TOO_BIG_FOR_Q (如果消息对于队列过大)
- MQRC_MSG_TOO_BIG_FOR_Q_MGR (如果消息对于队列管理器而言太大，但对于队列而言不是太大)

MaxMsgLength 属性的下限为零; 上限为 100 MB (104 857 600 字节)。

有关更多信息，请参阅 [MQPUT- BufferLength](#) 参数。

要确定此属性的值，请将 MQIA_MAX_MSG_LENGTH 选择器与 MQINQ 调用配合使用。

MaxQDepth (MQLONG)

这是定义的任何一次可以存在于队列中的物理消息数的上限。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

尝试将消息放入已包含 *MaxQDepth* 消息的队列失败，原因码为 MQRC_Q_FULL。

工作单元处理和消息分段都可能导致队列上的实际物理消息数超过 *MaxQDepth*。但是，这不会影响消息的可检索性; 可以使用 MQGET 调用来检索队列上的所有消息。

此属性的值为零或更大。上限由环境决定:

- 在 AIX, HP-UX, z/OS, Solaris, Linux 和 Windows 上, 该值不能超过 999 999 999。
- 在 IBM i 上, 该值不能超过 640 000。

注: 即使队列中的消息数少于 *MaxQDepth*, 可用于队列的存储空间也可能耗尽。

要确定此属性的值, 请将 MQIA_MAX_Q_DEPTH 选择器与 MQINQ 调用配合使用。

MsgDelivery 序列 (MQLONG)

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

这将确定 MQGET 调用将消息返回到应用程序的顺序:

MQMDS_FIFO

按 FIFO 顺序返回消息 (先进先出)。

MQGET 调用返回满足调用上指定的选择标准的 *first* 消息, 而不考虑消息的优先级。

MQMDS_PRIORITY

将按优先级顺序返回消息。

MQGET 调用返回满足调用上指定的选择标准的 *highest-priority* 消息。在每个优先级内, 将按 FIFO 顺序 (先入先出) 返回消息。

- 在 z/OS 上, 如果队列的 *IndexType* 为 MQIT_GROUP_ID, 那么 *MsgDeliverySequence* 属性指定将消息组返回到应用程序的顺序。返回组的特定顺序由每个组中第一条消息的位置或优先级确定。未定义队列上消息的物理顺序, 因为使用 MQGET 调用上的 MQGMO_LOGICAL_ORDER 选项对队列进行了优化以高效检索消息。
- 在 z/OS 上, 如果 *IndexType* 为 MQIT_GROUP_ID, *MsgDeliverySequence* 为 MQMDS_PRIORITY, 那么队列管理器将使用消息优先级 0 和 1 来优化按逻辑顺序检索消息。因此, 组中的第一条消息不得具有 0 或 1 的优先级; 如果具有 0 或 1 的优先级, 那么将处理该消息, 就好像它具有 2 的优先级一样。MQMD 结构中的 *Priority* 字段未更改。

如果在队列中有消息时更改了相关属性, 那么传递顺序如下所示:

- MQGET 调用返回消息的顺序由消息到达队列时对队列生效的 *MsgDeliverySequence* 和 *DefPriority* 属性的值确定:
 - 如果 *MsgDeliverySequence* 在消息到达时为 MQMDS_FIFO, 那么会将消息放在队列上, 就像其优先级为 *DefPriority* 一样。这不会影响消息的消息描述符中 *Priority* 字段的值; 该字段保留第一次放入消息时的值。
 - 如果在消息到达时 *MsgDeliverySequence* 为 MQMDS_PRIORITY, 那么会将消息放在队列中与消息描述符中的 *Priority* 字段所给定的优先级相应的位置。

如果在队列中存在消息时更改了 *MsgDeliverySequence* 属性的值, 那么不会更改队列中消息的顺序。

如果在队列上存在消息时更改了 *DefPriority* 属性的值, 那么不必按 FIFO 顺序传递消息, 即使 *MsgDeliverySequence* 属性设置为 MQMDS_FIFO; 将首先传递那些位于较高优先级的队列上的消息。

要确定此属性的值, 请将 MQIA_MSG_DELIVERY_SEQUENCE 选择器与 MQINQ 调用配合使用。

NonPersistentMessageClass (MQLONG)

非持久消息的可靠性目标。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

这指定废弃放入此队列的非持久消息的情况:

MQNPM_CLASS_NORMAL

非持久消息限于队列管理器会话的生存期; 如果队列管理器重新启动, 那么将废弃这些消息。这仅对非共享队列有效, 并且是缺省值。

MQNPM_CLASS_HIGH

队列管理器尝试在队列的生存期内保留非持久消息。发生故障时, 可能仍会丢失非持久消息。将对共享队列实施此值。

要确定此属性的值, 请将 MQIA_NPM_CLASS 选择器与 MQINQ 调用配合使用。

OpenInput 计数 (MQLONG)

这是当前有效的句柄数, 用于通过 MQGET 调用从队列中除去消息。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | | | |

这是本地队列管理器已知的此类句柄总数。如果队列是共享队列, 那么计数不包括在本地队列管理器所属的队列共享组中的其他队列管理器上为队列执行的输入打开操作。

计数包括为输入打开解析到此队列的别名队列的句柄。此计数不包括针对未包含输入的操作 (例如, 仅为浏览而打开的队列) 打开队列的句柄。

此属性的值在队列管理器运行时波动。

要确定此属性的值, 请将 MQIA_OPEN_INPUT_COUNT 选择器与 MQINQ 调用配合使用。

OpenOutput 计数 (MQLONG)

这是当前通过 MQPUT 调用将消息添加到队列的有效句柄数。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | | | |

它是本地队列管理器已知的此类句柄总数; 它不包括在远程队列管理器上对此队列执行的输出的打开数。如果队列是共享队列, 那么计数不包括在本地队列管理器所属的队列共享组中的其他队列管理器上为队列执行的输出打开操作。

该计数包括为输出打开解析到此队列的别名队列的句柄。计数不包括为不包含输出的操作 (例如, 仅为查询而打开的队列) 打开队列的句柄。

此属性的值在队列管理器运行时波动。

要确定此属性的值, 请将 MQIA_OPEN_OUTPUT_COUNT 选择器与 MQINQ 调用配合使用。

ProcessName (MQCHAR48)

这是在本地队列管理器上定义的进程对象的名称。进程对象标识可以为队列提供服务的程序。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

要确定此属性的值, 请将 MQCA_PROCESS_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_PROCESS_NAME_LENGTH 提供。

PropertyControl (MQLONG)

指定如何处理使用 MQGMO_PROPERTIES_AS_Q_DEF 选项从队列中检索的消息的消息属性。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | | |

该值为下列其中之一:

MQPROP_ALL

将消息传递到应用程序时，该消息的所有属性都包含在该消息中。除消息描述符（或扩展）中的属性之外的属性将放置在消息数据的一个或多个 MQRFH2 头中。如果提供了消息句柄，那么行为是返回消息句柄中的属性。

MQPROP_COMPATIBILITY

如果消息包含前缀为 mcd. 的属性，杰姆 乌斯河 或 mqext.，所有消息属性都以 MQRFH2 头交付到应用程序。否则，将废弃除消息描述符（或扩展）中包含的属性之外的所有消息属性，并且应用程序再也无法访问这些属性。这是缺省值；它允许应用程序在不加修改的情况下继续工作，该应用程序要求消息数据的 MQRFH2 头中具有与 JMS 相关的属性。如果提供了消息句柄，那么行为是返回消息句柄中的属性。

MQPROP_FORCE_MQRFH2

不管应用程序是否指定了消息句柄，始终都会在消息数据的 MQRFH2 头中返回属性。将忽略 MQGET 调用上 MQGMO 结构的 MsgHandle 字段中提供的有效消息句柄。无法通过该消息句柄访问消息的属性。

MQPROP_NONE

在将消息传递到应用程序之前，将从消息中除去消息的所有属性（消息描述符（或扩展）中的属性除外）。如果提供了消息句柄，那么行为是返回消息句柄中的属性。

此参数适用于本地队列，别名队列和模型队列。要确定其值，请将 MQIA_PROPERTY_CONTROL 选择器与 MQINQ 调用配合使用。

QDepthHigh 事件 (MQLONG)

这将控制是否生成 "队列深度高" 事件。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

"队列深度高" 事件指示应用程序已将消息放入队列，这已导致队列上的消息数大于或等于队列深度高阈值（请参阅 *QDepthHighLimit* 属性）。

注：此属性的值可以动态更改。

该值为下列其中之一：

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息，请参阅 [事件监视](#)。

要确定此属性的值，请将 MQIA_Q_DEPTH_HIGH_EVENT 选择器与 MQINQ 调用配合使用。

此属性在 z/OS 上受支持，但 MQINQ 调用无法用于确定其值。

QDepthHigh 限制 (MQLONG)

这是用于比较队列深度以生成 "队列深度上限" 事件的阈值。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

此事件指示应用程序已将消息放入队列中，并且这已导致队列中的消息数大于或等于队列深度高阈值。请参阅 [QDepthHigh 事件属性](#)。

该值以最大队列深度 (*MaxQDepth* 属性) 的百分比表示，并且大于或等于 0 且小于或等于 100。缺省值为 80。

要确定此属性的值，请将 MQIA_Q_DEPTH_HIGH_LIMIT 选择器与 MQINQ 调用配合使用。

此属性在 z/OS 上受支持，但 MQINQ 调用无法用于确定其值。

QDepthLow 事件 (MQLONG)

这将控制是否生成 "队列深度下限" 事件。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

"队列深度下限" 事件指示应用程序已从队列中检索消息，并且这已导致队列上的消息数小于或等于队列深度下限阈值 (请参阅 [QDepthLow 限制属性](#))。

注: 此属性的值可以动态更改。

该值为下列其中之一:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息，请参阅 [事件监视](#)。

要确定此属性的值，请将 MQIA_Q_DEPTH_LOW_EVENT 选择器与 MQINQ 调用配合使用。

此属性在 z/OS 上受支持，但 MQINQ 调用无法用于确定其值。

QDepthLow 限制 (MQLONG)

这是用于比较队列深度以生成 "队列深度下限" 事件的阈值。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

此事件指示应用程序已从队列中检索消息，并且这已导致队列上的消息数小于或等于队列深度下限阈值。请参阅 [QDepthLow 事件属性](#)。

该值以最大队列深度 (*MaxQDepth* 属性) 的百分比表示，并且大于或等于 0 且小于或等于 100。缺省值为 20。

要确定此属性的值，请将 MQIA_Q_DEPTH_LOW_LIMIT 选择器与 MQINQ 调用配合使用。

此属性在 z/OS 上受支持，但 MQINQ 调用无法用于确定其值。

QDepthMax 事件 (MQLONG)

这将控制是否生成 "队列已满" 事件。"队列已满" 事件指示已拒绝放入队列，因为队列已满，即队列深度已达到其最大值。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

注: 此属性的值可以动态更改。

该值为下列其中之一:

MQEVR_DISABLED

已禁用事件报告。

MQEVR_ENABLED

已启用事件报告。

有关事件的更多信息，请参阅 [事件监视](#)。

要确定此属性的值，请将 MQIA_Q_DEPTH_MAX_EVENT 选择器与 MQINQ 调用配合使用。

此属性在 z/OS 上受支持，但 MQINQ 调用无法用于确定其值。

QDesc (MQCHAR64)

将此字段用于描述性注释。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | X | X |

该字段的内容对队列管理器没有意义，但队列管理器可能要求该字段仅包含可显示的字符。它不能包含任何空字符；如有必要，将用空格填充到右边。在 DBCS 安装中，该字段可包含 DBCS 字符 (最大字段长度为 64 字节)。

注: 如果此字段包含不在队列管理器的字符集中的字符 (如 *CodedCharSetId* 队列管理器属性所定义)，那么如果将此字段发送到另一个队列管理器，那么这些字符可能转换不正确。

要确定此属性的值，请将 MQCA_Q_DESC 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_DESC_LENGTH 提供。

QName (MQCHAR48)

这是在本地队列管理器上定义的队列的名称。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | X |

队列管理器上定义的所有队列共享同一个队列名称空间。因此，MQQT_LOCAL 队列和 MQQT_ALIAS 队列不能同名。

要确定此属性的值，请将 MQCA_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

QServiceInterval (MQLONG)

这是用于比较以生成 "服务时间间隔高" 和 "服务时间间隔正常" 事件的服务时间间隔。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

请参阅 *QServiceInterval* 事件属性。

该值以毫秒为单位，大于或等于零，小于或等于 999 999 999 999。

要确定此属性的值，请将 MQIA_Q_SERVICE_INTERVAL 选择器与 MQINQ 调用配合使用。

此属性在 z/OS 上受支持，但 MQINQ 调用无法用于确定其值。

QServiceInterval 事件 (MQLONG)

这将控制是生成 "服务时间间隔高" 还是 "服务时间间隔正常" 事件。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

- 当检查指示至少在 *QServiceInterval* 属性指示的时间内未从队列中检索消息时，将生成 "服务时间间隔高" 事件。
- 当检查指示已在 *QServiceInterval* 属性指示的时间内从队列中检索消息时，将生成 "服务时间间隔正常" 事件。

注: 此属性的值可以动态更改。

该值为下列其中之一：

MQQSIE_HIGH

队列服务时间间隔高事件已启用。

- 队列服务时间间隔高事件已 **启用**，并且
- 队列服务时间间隔正常事件 **已禁用**。

MQQSIE_OK

队列服务时间间隔正常事件已启用。

- 队列服务时间间隔高事件 **已禁用** 并且
- 队列服务时间间隔正常事件已 **启用**。

MQQSIE_NONE

未启用队列服务时间间隔事件。

- 队列服务时间间隔高事件 **已禁用** 并且
- 队列服务时间间隔正常事件也 **已禁用**。

对于共享队列，将忽略此属性的值；将采用值 MQQSIE_NONE。

有关事件的更多信息，请参阅 [事件监视](#)。

要确定此属性的值，请将 MQIA_Q_SERVICE_INTERVAL_EVENT 选择器与 MQINQ 调用配合使用。

在 z/OS 上，无法使用 MQINQ 调用来确定此属性的值。

QSGDisp (MQLONG)

这指定队列的处置。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | |

该值为下列其中之一：

MQQSGD_Q_MGR

对象具有队列管理器处置。这意味着对象定义仅对本地队列管理器是已知的；该定义对队列共享组中的其他队列管理器是未知的。

队列共享组中的每个队列管理器都可以具有与当前对象具有相同名称和类型的对象，但这些对象是单独的对象，并且它们之间没有关联。它们的属性并不被约束为彼此相同。

MQQSGD_COPY

该对象是共享存储库中存在的主对象定义的本地副本。队列共享组中的每个队列管理器都可以有自己的对象副本。最初，所有副本都具有相同的属性，但通过使用 MQSC 命令，您可以更改每个副本，以使其属性与其他副本的属性不同。当更改共享存储库中的主定义时，将再同步这些副本的属性。

MQQSGD_SHARED

对象具有共享处置。这意味着共享存储库中存在队列共享组中所有队列管理器已知的对象的单个实例。当组中的队列管理器访问对象时，它将访问对象的单个共享实例。

要确定此属性的值，请将 MQIA_QSG_DISP 选择器与 MQINQ 调用配合使用。

此属性仅在 z/OS 上受支持。

QueueAccounting (MQLONG)

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | X | |

这将控制队列的记帐数据收集。对于要为此队列收集的记帐数据，还必须使用 QMGR 属性 ACCTQ 或 MQCONNX 调用上 MQCNO 结构中的 "选项" 字段来启用此连接的记帐数据。

此属性有下列某个值：

MQMON_Q_MGR

根据 QMGR 属性 ACCTQ 的设置收集此队列的记帐数据。这是缺省设置。

MQMON_OFF

请勿收集此队列的记帐数据。

MQMON_ON

收集此队列的记帐数据。

要确定此属性的值，请将 MQIA_ACCOUNTING_Q 选择器与 MQINQ 调用配合使用。

QueueMonitoring (MQLONG)

控制队列联机监视数据的收集。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

该值为下列其中之一：

MQMON_Q_MGR

根据 *QueueMonitoring* 队列管理器属性的设置收集监视数据。这是缺省值。

MQMON_OFF

对此队列关闭联机监视数据收集。

MQMON_LOW

如果 *QueueMonitoring* 队列管理器属性的值不是 MQMON_NONE，那么将开启联机监视数据收集，并且此队列的数据收集速率较低。

MQMON_MEDIUM

如果 *QueueMonitoring* 队列管理器属性的值不是 MQMON_NONE，那么将开启联机监视数据收集，并且此队列的数据收集速率适中。

MQMON_HIGH

如果 *QueueMonitoring* 队列管理器属性的值不是 MQMON_NONE，那么将开启联机监视数据收集，并且此队列的数据收集速率很高。

要确定此属性的值，请将 MQIA_MONITORING_Q 选择器与 MQINQ 调用配合使用。

QueueStatistics (MQCHAR12)

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | X | X | |

这将控制队列的统计数据收集。

此属性有下列某个值：

MQMON_Q_MGR

将根据 QMGR 属性 STATQ 的设置来收集此队列的记帐数据。这是缺省设置。

MQMON_OFF

关闭此队列的统计信息数据收集。

MQMON_ON

打开此队列的统计数据收集。

QType (MQLONG)

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | X |

这是队列的类型; 它具有下列其中一个值:

MQQT_ALIAS

别名队列定义。

MQQT_CLUSTER

集群队列。

MQQT_LOCAL

本地队列。

MQQT_REMOTE

远程队列的本地定义。

要确定此属性的值, 请将 MQQA_Q_TYPE 选择器与 MQINQ 调用配合使用。

RemoteQMgr 名称 (MQCHAR48)

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| | | | X | |

这是定义了队列 *RemoteQName* 的远程队列管理器的名称。如果 *RemoteQName* 队列的 *QSGDisp* 值为 MQQSGD_COPY 或 MQQSGD_SHARED, 那么 *RemoteQMgrName* 可以是拥有 *RemoteQName* 的队列共享组的名称。

如果应用程序打开远程队列的本地定义, 那么 *RemoteQMgrName* 不得为空, 并且不得为本地队列管理器的名称。如果 *XmitQName* 为空, 那么将使用与 *RemoteQMgrName* 同名的本地队列作为传输队列。如果没有名称为 *RemoteQMgrName* 的队列, 那么将使用由 *DefXmitQName* queue-manager 属性标识的队列。

如果此定义用于队列管理器别名, 那么 *RemoteQMgrName* 是要创建别名的队列管理器的名称。它可以是本地队列管理器的名称。否则, 如果 *XmitQName* 在打开时为空白, 那么必须存在名称与 *RemoteQMgrName* 相同的本地队列; 此队列将用作传输队列。

如果此定义用于应答别名, 那么此名称是要作为 *ReplyToQMgr* 的队列管理器的名称。

注: 在创建或修改队列定义时, 不会对此属性指定的值执行验证。

要确定此属性的值, 请将 MQCA_REMOTE_Q_MGR_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_MGR_NAME_LENGTH 指定。

RemoteQName (MQCHAR48)

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| | | | X | |

这是在远程队列管理器 *RemoteQMgrName* 上已知的队列名称。

如果应用程序打开远程队列的本地定义, 那么当打开时 *RemoteQName* 不得为空。

如果此定义用于队列管理器别名定义, 那么当打开时 *RemoteQName* 必须为空。

如果定义用于应答别名, 那么此名称是要作为 *ReplyToQ* 的队列的名称。

注: 在创建或修改队列定义时, 不会对此属性指定的值执行验证。

要确定此属性的值, 请将 MQCA_REMOTE_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

RetentionInterval (MQLONG)

这是保留队列的时间段。经过此时间后, 该队列符合删除条件。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

时间以小时为单位，从创建队列的日期和时间开始计算。队列的创建日期和时间记录在 *CreationDate* 和 *CreationTime* 属性中。

提供此信息是为了使清理应用程序或操作员能够识别和删除不再需要的队列。

注：队列管理器从不执行任何操作来根据此属性删除队列，也不阻止删除具有未到期的保留时间间隔的队列；用户负责执行任何必需的操作。

使用现实的保留时间间隔来防止累积永久动态队列 (请参阅 [DefinitionType](#) 属性)。但是，此属性也可以与预定义队列配合使用。

要确定此属性的值，请将 MQIA_RETENTION_INTERVAL 选择器与 MQINQ 调用配合使用。

作用域 (MQLONG)

这控制此队列的条目是否也存在于单元目录中。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | | X | X | |

单元目录由可安装的名称服务提供。该值为下列其中之一：

MQSCO_Q_MGR

队列定义具有队列管理器作用域: 队列的定义不会超出拥有该队列的队列管理器的范围。要打开队列以从其他某个队列管理器输出，必须指定拥有队列管理器的名称，或者另一个队列管理器必须具有队列的本地定义。

MQSCO_CELL

队列定义具有单元作用域: 队列定义也放置在单元目录中，可供单元中的所有队列管理器使用。可以通过指定队列的名称为单元中任何队列管理器的输出打开队列; 无需指定拥有该队列的队列管理器的名称。但是，该队列定义对于单元中还具有具有该名称的队列的本地定义的任何队列管理器都不可用，因为本地定义优先。

单元目录由可安装的名称服务提供。

模型和动态队列不能具有单元作用域。

仅当配置了支持单元目录的名称服务时，此值才有效。

要确定此属性的值，请将 MQIA_SCOPE 选择器与 MQINQ 调用配合使用。

对此属性的支持受以下限制:

- 在 IBM i 上，支持此属性，但只有 MQSCO_Q_MGR 有效。
- 在 z/OS 上，不支持此属性。

可共享性 (MQLONG)

这指示是否可以同时多次打开队列以进行输入。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

该值为下列其中之一：

MQQA_SHAREABLE

队列可共享。

允许使用 MQOO_INPUT_SHARED 选项进行多次打开。

MQQA_NOT_SHAREABLE

队列不可共享。

使用 MQOO_INPUT_SHARED 选项的 MQOPEN 调用被视为 MQOO_INPUT_EXCLUSIVE。

要确定此属性的值，请将 MQIA_SHAREABILITY 选择器与 MQINQ 调用配合使用。

StorageClass (MQCHAR8)

这是用户定义的名称，用于定义用于存放队列的物理存储器。实际上，仅当需要将消息从其内存缓冲区中分页时，才会将该消息写入磁盘。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

要确定此属性的值，请将 MQCA_STORAGE_CLASS 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_STORAGE_CLASS_LENGTH 指定。

此属性仅在 z/OS 上受支持。

TriggerControl (MQLONG)

这将控制是否将触发器消息写入启动队列以启动应用程序来为队列提供服务。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

这是下列状态之一：

MQTC_OFF

不会为此队列写入任何触发器消息。在这种情况下，*TriggerType* 的值不相关。

MQTC_ON

当发生相应的触发器事件时，将为此队列写入触发器消息。

要确定此属性的值，请将 MQIA_TRIGGER_CONTROL 选择器与 MQINQ 调用配合使用。要更改此属性的值，请使用 MQSET 调用。

TriggerData (MQCHAR64)

当到达此队列的消息导致将触发器消息写入启动队列时，这是队列管理器插入到触发器消息中的自由格式数据。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

此数据的内容对队列管理器没有任何意义。这对于处理启动队列的触发器监视器应用程序或触发器监视器启动的应用程序都有意义。

字符串不得包含任何空值。如有必要，将用空格填充到右侧。

要确定此属性的值，请将 MQCA_TRIGGER_DATA 选择器与 MQINQ 调用配合使用。要更改此属性的值，请使用 MQSET 调用。此属性的长度由 MQ_TRIGGER_DATA_LENGTH 指定。

TriggerDepth (MQLONG)

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

这是在写入触发器消息之前必须位于队列上的优先级为 *TriggerMsgPriority* 或更高的消息数。当 *TriggerType* 设置为 MQTT_DEPTH 时适用。*TriggerDepth* 的值为 1 或更大的值。否则不使用此属性。

要确定此属性的值，请将 MQIA_TRIGGER_DEPTH 选择器与 MQINQ 调用配合使用。要更改此属性的值，请使用 MQSET 调用。

TriggerMsg 优先级 (MQLONG)

这是消息优先级, 在此优先级下, 消息不会生成触发器消息 (即, 队列管理器在决定是否生成触发器消息时将忽略这些消息)。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

TriggerMsgPriority 可以在范围 0 (最低) 到 *MaxPriority* (最高; 请参阅 [MaxPriority 属性](#)) 之间; 值为零会导致所有消息生成触发器消息。

要确定此属性的值, 请将 MQIA_TRIGGER_MSG_PRIORITY 选择器与 MQINQ 调用配合使用。要更改此属性的值, 请使用 MQSET 调用。

TriggerType (MQLONG)

这将控制由于消息到达此队列而写入触发器消息的条件。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

它具有下列其中一个值:

MQTT_NONE

不会由于此队列中的消息而写入任何触发器消息。这与将 *TriggerControl* 设置为 MQTC_OFF 具有相同的效果。

MQTT_FIRST

每当队列上优先级为 *TriggerMsgPriority* 或更大的消息数从 0 更改为 1 时, 都会写入触发器消息。

MQTT EVERY

每当优先级为 *TriggerMsgPriority* 或更高的消息到达队列时, 都会写入触发器消息。

MQTT_DEPTH

每当队列上优先级 *TriggerMsgPriority* 或更大的消息数等于或超过 *TriggerDepth* 时, 都会写入触发器消息。写入触发器消息后, *TriggerControl* 将设置为 MQTC_OFF 以防止进一步触发, 直到再次显式开启该消息为止。

要确定此属性的值, 请将 MQIA_TRIGGER_TYPE 选择器与 MQINQ 调用配合使用。要更改此属性的值, 请使用 MQSET 调用。

用法 (MQLONG)

这指示队列的用途。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| X | X | | | |

该值为下列其中之一:

MQUS_NORMAL

这是应用程序在放入和获取消息时使用的队列; 该队列不是传输队列。

MQUS_TRANSMISSION

这是用于保存发往远程队列管理器的消息的队列。当应用程序向远程队列发送消息时, 本地队列管理器以特殊格式将消息临时存储在相应的传输队列上。然后, 消息通道代理程序从传输队列中读取消息, 并将消息传输到远程队列管理器。有关传输队列的更多信息, 请参阅 [定义传输队列](#)。

只有特权应用程序才能为 MQOO_OUTPUT 打开传输队列, 以将消息直接放在其上。通常, 只有实用程序应用程序执行此操作。确保消息数据格式正确 (请参阅第 540 页的『MQXQH-传输队列头』), 否则在传输过程中可能会发生错误。除非指定了其中一个 MQPMO_*_CONTEXT 上下文选项, 否则不会传递或设置上下文。

要确定此属性的值, 请将 MQIA_USAGE 选择器与 MQINQ 调用配合使用。

XmitQName (MQCHAR48)

这是传输队列名称。如果针对远程队列或队列管理器别名定义进行打开时此属性为非空白，那么它指定要用于转发消息的本地传输队列的名称。

| 本地 | 模型 | 别名 | 远程 | 集群 |
|----|----|----|----|----|
| | | | X | |

如果 *XmitQName* 为空，那么名称与 *RemoteQMGrName* 相同的本地队列将用作传输队列。如果没有名称为 *RemoteQMGrName* 的队列，那么将使用由 *DefXmitQName* queue-manager 属性标识的队列。

如果定义用作队列管理器别名，并且 *RemoteQMGrName* 是本地队列管理器的名称，那么将忽略此属性。如果此定义用作应答队列别名定义，那么也忽略它。

要确定此属性的值，请将 MQCA_XMIT_Q_NAME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_Q_NAME_LENGTH 提供。

名称列表的属性

下表概述了特定于名称列表的属性。这些属性按字母顺序进行描述。

所有 WebSphere MQ 系统以及连接到这些系统的 WebSphere MQ MQI 客户机都支持名称列表。

注：本节中显示的属性名称是用于 MQINQ 和 MQSET 调用的描述性名称；这些名称与 PCF 命令相同。当 MQSC 命令用于定义，改变或显示属性时，将使用备用短名称；有关更多信息，请参阅脚本 (MQSC) 命令。

| 属性 | 描述 |
|-----------------------|------------------------|
| <u>AlterationDate</u> | 上次更改定义日期 |
| <u>AlterationTime</u> | 上次更改定义时间 |
| <u>NameCount</u> | 名称列表中的名称数 |
| <u>NamelistDesc</u> | 名称列表描述 |
| <u>NamelistName</u> | 名称列表名称 |
| 名称 | <i>NameCount</i> 名称的列表 |
| <u>NamelistType</u> | 名称列表类型 |
| <u>QSGDisp</u> | 队列共享组处置 |

AlterationDate (MQCHAR12)

这是上次更改定义的日期。日期的格式为 YYYY-MM-DD，用两个尾部空格填充，使长度为 12 个字节。

要确定此属性的值，请将 MQCA_ALTERATION_DATE 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_DATE_LENGTH 提供。

AlterationTime (MQCHAR8)

这是上次更改定义的时间。时间的格式为 HH.MM.SS。

要确定此属性的值，请将 MQCA_ALTERATION_TIME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_TIME_LENGTH 提供。

NameCount (MQLONG)

这是名称列表中的名称数。大于或等于零。定义了以下值：

MQNC_MAX_NAMELIST_NAME_COUNT

名称列表中的最大名称数。

要确定此属性的值，请将 MQIA_NAME_COUNT 选择器与 MQINQ 调用配合使用。

NamelistDesc (MQCHAR64)

将此字段用于描述性注释; 其值由定义过程建立。 该字段的内容对队列管理器没有意义, 但队列管理器可能要求该字段仅包含可显示的字符。 它不能包含任何空字符; 如有必要, 将用空格填充到右边。 在 DBCS 安装中, 此字段可包含 DBCS 字符 (最大字段长度为 64 字节)。

注: 如果此字段包含不在队列管理器的字符集中的字符 (如 `CodedCharSetId` 队列管理器属性所定义), 那么如果将此字段发送到另一个队列管理器, 那么这些字符可能转换不正确。

要确定此属性的值, 请将 `MQCA_NAMELIST_DESC` 选择器与 `MQINQ` 调用配合使用。

此属性的长度由 `MQ_NAMELIST_DESC_LENGTH` 提供。

NamelistName (MQCHAR48)

这是在本地队列管理器上定义的名称列表的名称。 有关名称列表名称的更多信息, 请参阅 [其他对象名称](#) 部分。

每个名称列表都具有与属于队列管理器的其他名称列表的名称不同的名称, 但可能会复制不同类型 (例如, 队列) 的其他队列管理器对象的名称。

要确定此属性的值, 请将 `MQCA_NAMELIST_NAME` 选择器与 `MQINQ` 调用配合使用。

此属性的长度由 `MQ_NAMELIST_NAME_LENGTH` 提供。

NamelistType (MQLONG)

这指定名称列表中名称的性质, 并指示名称列表的使用方式。 它是下列其中一个值:

MQNT_NONE

未指定类型的名称列表。

MQNT_Q

包含队列名称的名称列表。

MQNT_CLUSTER

包含集群名称的名称列表。

MQNT_AUTH_INFO

包含 authentication-information 对象的名称的名称列表。

要确定此属性的值, 请将 `MQIA_NAMELIST_TYPE` 选择器与 `MQINQ` 调用配合使用。

此属性仅在 z/OS 上受支持。

名称 (MQCHAR48xNameCount)

这是 *NameCount* 名称的列表, 其中每个名称都是对本地队列管理器定义的对象名称。 有关对象名的更多信息, 请参阅 [用于命名 IBM WebSphere MQ 对象的规则](#)。

要确定此属性的值, 请将 `MQCA_NAMES` 选择器与 `MQINQ` 调用配合使用。

列表中每个名称的长度由 `MQ_OBJECT_NAME_LENGTH` 提供。

QSGDisp (MQLONG)

这指定名称列表的处置。 该值为下列其中之一:

MQQSGD_Q_MGR

对象具有队列管理器处置: 对象定义仅对本地队列管理器已知; 队列共享组中的其他队列管理器不知道该定义。

队列共享组中的每个队列管理器都可以具有与当前对象具有相同名称和类型的对象, 但这些对象是单独的对象, 并且它们之间没有关联。 它们的属性并不被约束为彼此相同。

MQQSGD_COPY

该对象是共享存储库中存在的主对象定义的本地副本。 队列共享组中的每个队列管理器都可以有自己的对象副本。 最初, 所有副本都具有相同的属性, 但您可以使用 `MQSC` 命令来变更每个副本, 以使其属性与其他副本的属性不同。 当更改共享存储库中的主定义时, 将再同步这些副本的属性。

要确定此属性的值, 请将 `MQIA_QSG_DISP` 选择器与 `MQINQ` 调用配合使用。

此属性仅在 z/OS 上受支持。

进程定义的属性

下表概述了特定于进程定义的属性。这些属性按字母顺序进行描述。

注: 此部分中的属性名称是用于 MQINQ 和 MQSET 调用的描述性名称; 这些名称与 PCF 命令相同。当 MQSC 命令用于定义, 改变或显示属性时, 将使用备用短名称; 有关更多信息, 请参阅 脚本 (MQSC) 命令。

| 属性 | 描述 |
|--------------------------------|----------|
| AlterationDate | 上次更改定义日期 |
| AlterationTime | 上次更改定义时间 |
| ApplId | 应用程序标识 |
| ApplType | 应用程序类型 |
| EnvData | 环境数据 |
| ProcessDesc | 流程描述 |
| ProcessName | 进程名称 |
| QSGDisp | 队列共享组处置 |
| UserData | 用户数据 |

AlterationDate (MQCHAR12)

这是上次更改定义的日期。日期的格式为 YYYY-MM-DD, 用两个尾部空格填充, 使长度为 12 个字节。

要确定此属性的值, 请将 MQCA_ALTERATION_DATE 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_DATE_LENGTH 提供。

AlterationTime (MQCHAR8)

这是上次更改定义的时间。时间的格式为 HH.MM.SS。

要确定此属性的值, 请将 MQCA_ALTERATION_TIME 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_TIME_LENGTH 提供。

ApplId (MQCHAR256)

这是一个字符串, 用于标识要启动的应用程序。此信息供处理启动队列上的消息的触发器监视器应用程序使用; 此信息将作为触发器消息的一部分发送到启动队列。

ApplId 的含义由 trigger-monitor 应用程序确定。WebSphere MQ 提供的触发器监视器要求 *ApplId* 是可执行程序的名称。以下说明适用于所指示的环境:

- 在 z/OS 上, *ApplId* 必须为:
 - 使用 CICS 触发器监视器事务 CKTI 启动的应用程序的 CICS 事务标识
 - 使用 IMS 触发器监视器 CSQQTRMN 启动的应用程序的 IMS 事务标识
- 在 Windows 系统上, 可以使用驱动器和目录路径作为程序名的前缀。
- 在 UNIX 系统上, 可以使用目录路径作为程序名的前缀。

字符串不能包含任何空值。如有必要, 将用空格填充到右侧。

要确定此属性的值, 请将 MQCA_APPL_ID 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_PROCESS_APPL_ID_LENGTH 提供。

ApplType (MQLONG)

这标识要启动以响应接收触发器消息的程序的性质。此信息供处理启动队列上的消息的触发器监视器应用程序使用; 此信息将作为触发器消息的一部分发送到启动队列。

AppType 可以具有任何值，但对于标准类型，建议使用以下值；通过 MQAT_USER_LAST 将用户定义的应用程序类型限制为 MQAT_USER_FIRST 范围内的值：

MQAT_AIX

AIX 应用程序 (与 MQAT_UNIX 的值相同)。

MQAT_BATCH

批处理应用程序

MQAT_BROKER

代理应用程序

MQAT_CICS

CICS 事务。

MQAT_CICS_BRIDGE

CICS 网桥应用程序。

MQAT_CICS_VSE

CICS/VSE 事务。

MQAT_DOS

PC DOS 上的 WebSphere MQ MQI 客户机应用程序。

MQAT_IMS

IMS 应用程序。

MQAT_IMS_BRIDGE

IMS 网桥应用程序。

MQAT_JAVA

Java 应用程序。

MQAT_MVS

MVS 或 TSO 应用程序 (与 MQAT_ZOS 值相同)。

MQAT_NOTES_AGENT

Lotus Notes 代理程序应用程序。

MQAT_NSK

HP Integrity NonStop Server 应用程序。

MQAT_OS390

OS/390 应用程序 (与 MQAT_ZOS 的值相同)。

MQAT_OS400

IBM i 应用程序。

MQAT_RRS_BATCH

RRS 批处理应用程序。

MQAT_UNIX

UNIX 应用程序。

MQAT_UNKNOWN

未知类型的应用程序。

MQAT_USER

用户应用程序。

MQAT_VOS

Stratus VOS 应用程序。

MQAT_WINDOWS

16 位 Windows 应用程序。

MQAT_WINDOWS_NT

32 位 Windows 应用程序。

MQAT_WLM

z/OS 工作负载管理器应用程序。

MQAT_XCF

XCF。

MQAT_ZOS

z/OS 应用程序。

MQAT_USER_FIRST

用户定义的应用程序类型的最小值。

MQAT_USER_LAST

用户定义的应用程序类型的最大值。

要确定此属性的值，请将 MQIA_APPL_TYPE 选择器与 MQINQ 调用配合使用。

EnvData (MQCHAR128)

这是一个字符串，其中包含与要启动的应用程序相关的环境信息。此信息供处理启动队列上的消息的触发器监视器应用程序使用；此信息将作为触发器消息的一部分发送到启动队列。

EnvData 的含义由 trigger-monitor 应用程序确定。WebSphere MQ 提供的触发器监视器将 *EnvData* 附加到传递到已启动应用程序的参数列表。参数列表由 MQTMC2 结构组成，后跟一个空格，后跟 *EnvData* (除了尾部空格)。以下说明适用于所指示的环境：

- 在 z/OS 上：
 - WebSphere MQ 提供的触发器监视器应用程序不使用 *EnvData*。
 - 如果 ApplType 为 MQAT_WLM，那么可以在 *EnvData* 中为工作信息头 (MQWIH) 中的 ServiceName 和 ServiceStep 字段提供缺省值。
- 在 UNIX 系统上，可以将 *EnvData* 设置为 & 字符以在后台运行已启动的应用程序。

字符串不能包含任何空值。如有必要，将用空格填充到右侧。

要确定此属性的值，请将 MQCA_ENV_DATA 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_PROCESS_ENV_DATA_LENGTH 指定。

ProcessDesc (MQCHAR64)

将此字段用于描述性注释。该字段的内容对队列管理器没有意义，但队列管理器可能要求该字段仅包含可显示的字符。它不能包含任何空字符；如有必要，将用空格填充到右边。在 DBCS 安装中，该字段可包含 DBCS 字符 (最大字段长度为 64 字节)。

注：如果此字段包含不在队列管理器的字符集中的字符 (如 *CodedCharSetId* 队列管理器属性所定义)，那么如果将此字段发送到另一个队列管理器，那么这些字符可能转换不正确。

要确定此属性的值，请将 MQCA_PROCESS_DESC 选择器与 MQINQ 调用配合使用。

此属性的长度由 MQ_PROCESS_DESC_LENGTH 提供。

ProcessName (MQCHAR48)

这是在本地队列管理器上定义的进程定义的名称。

每个进程定义都具有与属于队列管理器的其他进程定义的名称不同的名称。但是，进程定义的名称可能与不同类型 (例如，队列) 的其他队列管理器对象的名称相同。

要确定此属性的值，请将 MQCA_PROCESS_NAME 选择器与 MQINQ 调用配合使用。

此属性的长度由 MQ_PROCESS_NAME_LENGTH 提供。

QSGDisp (MQLONG)

这指定进程定义的处置。该值为下列其中之一：

MQQSGD_Q_MGR

对象具有队列管理器处置：对象定义仅对本地队列管理器已知；队列共享组中的其他队列管理器不知道该定义。

队列共享组中的每个队列管理器都可以具有与当前对象具有相同名称和类型的对象，但这些对象是单独的对象，并且它们之间没有关联。它们的属性并不被约束为彼此相同。

MQQSGD_COPY

该对象是共享存储库中存在的主对象定义的本地副本。队列共享组中的每个队列管理器都可以有自己的对象副本。最初，所有副本都具有相同的属性，但您可以使用 MQSC 命令来变更每个副本，以使其属性与其他副本的属性不同。当更改共享存储库中的主定义时，将再同步这些副本的属性。

要确定此属性的值，请将 MQIA_QSG_DISP 选择器与 MQINQ 调用配合使用。

此属性仅在 z/OS 上受支持。

UserData (MQCHAR128)

UserData 是一个字符串，其中包含与要启动的应用程序相关的用户信息。此信息供处理启动队列上的消息的触发器监视器应用程序或由触发器监视器启动的应用程序使用。该信息将作为触发器消息的一部分发送到启动队列。

UserData 的含义由 trigger-monitor 应用程序确定。WebSphere MQ 提供的触发器监视器将 UserData 作为参数列表的一部分传递到已启动的应用程序。参数列表由 MQTMC2 结构 (包含 UserData) 组成，后跟一个空格，后跟 EnvData (除去了尾部空格)。

字符串不能包含任何空值。如有必要，将用空格填充到右侧。对于 Microsoft Windows，如果要将进程定义传递到 **runmqtrm**，那么字符串不得包含双引号。

要确定此属性的值，请将 MQCA_USER_DATA 选择器与 MQINQ 调用配合使用。此属性的长度由 MQ_PROCESS_USER_DATA_LENGTH 指定。

返回码

对于每个 WebSphere MQ 消息队列接口 (MQI) 和 WebSphere MQ 管理接口 (MQAI) 调用，队列管理器或出口例程会返回 **完成** 代码和 **原因** 代码，以指示调用成功或失败。

应用程序不得依赖于按特定顺序检查的错误，除非有特别说明。如果一个调用可能产生多个完成代码或原因码，那么报告的特定错误取决于实现。

在 WebSphere MQ API 调用之后检查成功完成的应用程序必须始终检查完成代码。不要根据原因码的值来假定完成代码值。

完成代码

完成代码参数 (*CompCode*) 允许调用者快速查看调用是成功完成，部分完成还是失败。以下是完成代码的列表，其详细信息比调用描述中提供的详细信息更多：

MQCC_OK

调用圆满完成；已设置所有输出参数。在此情况下，*Reason* 参数的值始终为 MQRC_NONE。

MQCC_WARNING

部分完成了调用。除 *CompCode* 和 *Reason* 输出参数外，可能还设置了一些输出参数。*Reason* 参数提供有关部分完成的其他信息。

MQCC_FAILED

调用的处理未完成。队列管理器的状态保持不变，但有特别说明的情况除外。已设置 *CompCode* 和 *Reason* 输出参数；其他参数保持不变，除非另有说明。

原因可能是应用程序中的故障，也可能是程序外部的某些情况 (例如，用户的权限可能已被撤销) 造成的。*Reason* 参数提供有关错误的其他信息。

原因码

原因码参数 (*Reason*) 限定完成代码参数 (*CompCode*)。

如果没有需要报告的特殊原因，那么将返回 MQRC_NONE。成功调用将返回 MQCC_OK 和 MQRC_NONE。

如果完成代码为 MQCC_WARNING 或 MQCC_FAILED，那么队列管理器将始终报告合理的原因；每个调用描述下都会提供详细信息。

当用户出口例程设置完成代码和原因时，它们必须遵守这些规则。此外，用户出口定义的任何特殊原因值都必须小于零，以确保它们与队列管理器定义的值不冲突。出口可以设置队列管理器已定义的原因 (如果适用)。

原因码还会出现在：

- MQDLH 结构的 *Reason* 字段
- MQMD 结构的 *Feedback* 字段

有关原因码的完整描述，请参阅 [原因码](#)。

验证 MQI 选项的规则

本部分列出了从 MQOPEN，MQPUT，MQPUT1，MQGET，MQCLOSE 或 MQSUB 调用生成 MQRC_OPTIONS_ERROR 原因码的情境。

MQOPEN 调用

对于 MQOPEN 调用的选项：

- 必须至少指定下列其中一个：
 - MQOO_BROWSE
 - MQOO_INPUT_EXCLUSIVE¹
 - MQOO_INPUT_SHARED¹
 - MQOO_INPUT_AS_Q_DEF¹
 - MQOO_INQUIRE
 - MQOO_OUTPUT
 - MQOO_SET
 - MQOO_BIND_ON_OPEN²
 - MQOO_BIND_NOT_FIXED²
 - MQOO_BIND_ON_GROUP²
 - MQOO_BIND_AS_Q_DEF²
- 仅允许一个以下项：
 - MQOO_READ_AHEAD
 - MQOO_NO_READ_AHEAD
 - MQOO_READ_AHEAD_AS_Q_DEF

1. 仅允许一个以下项：

- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_SHARED
- MQOO_INPUT_AS_Q_DEF

2. 仅允许一个以下项：

- MQOO_BIND_ON_OPEN
- MQOO_BIND_NOT_FIXED
- MQOO_BIND_ON_GROUP
- MQOO_BIND_AS_Q_DEF

注: 上面列出的选项是互斥的。但是, 由于 MQOO_BIND_AS_Q_DEF 的值为零, 因此将其与其他两个绑定选项中的任何一个一起指定不会导致原因码 MQRC_OPTIONS_ERROR。提供 MQOO_BIND_AS_Q_DEF 以帮助程序文档。

- 如果指定了 MQOO_SAVE_ALL_CONTEXT, 那么还必须指定其中一个 MQOO_INPUT_* 选项。
- 如果指定了其中一个 MQOO_SET_*_CONTEXT 或 MQOO_PASS_*_CONTEXT 选项, 那么还必须指定 MQOO_OUTPUT。
- 如果指定了 MQOO_CO_OP, 那么还必须指定 MQOO_BROWSE
- 如果指定了 MQOO_NO_多点广播, 那么还必须指定 MQOO_OUTPUT。

MQPUT 调用

对于 put-message 选项:

- 不允许 MQPMO_SYNCPOINT 和 MQPMO_NO_SYNCPOINT 的组合。
- 仅允许一个 以下项:
 - MQPMO_DEFAULT_CONTEXT
 - MQPMO_NO_CONTEXT
 - MQPMO_PASS_ALL_CONTEXT
 - MQPMO_PASS_IDENTITY_CONTEXT
 - MQPMO_SET_ALL_CONTEXT
 - MQPMO_SET_IDENTITY_CONTEXT
- 仅允许一个 以下项:
 - MQPMO_ASYNC_RESPONSE
 - MQPMO_SYNC_RESPONSE
 - MQPMO_RESPONSE_AS_TOPIC_DEF
 - MQPMO_RESPONSE_AS_Q_DEF
- 不允许 MQPMO_ALTERNATE_USER_AUTHORITY (仅在 MQPUT1 调用上有效)。

MQPUT1 调用

对于 put-message 选项, 规则与 MQPUT 调用相同, 但以下规则除外:

- 允许 MQPMO_ALTERNATE_USER_AUTHORITY。
- 不允许 MQPMO_LOGICAL_ORDER。

MQGET 调用

对于 get-message 选项:

- 仅允许一个 以下项:
 - MQGMO_NO_SYNCPOINT
 - MQGMO_SYNCPOINT
 - MQGMO_SYNCPOINT_IF_PERSISTENT
- 仅允许一个 以下项:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_MSG_UNDER_CURSOR
- 以下任何一项都不允许 MQGMO_SYNCPOINT:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_UNLOCK
- 下列任何一项都不允许 MQGMO_SYNCPOINT_IF_PERSISTENT:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_COMPLETE_MSG
 - MQGMO_UNLOCK
- MQGMO_MARK_SKIP_BACKOUT 需要指定 MQGMO_SYNCPOINT。
- 不允许 MQGMO_WAIT 与 MQGMO_SET_SIGNAL 的组合。
- 如果指定了 MQGMO_LOCK , 那么还必须指定下列其中一项:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
- 如果指定了 MQGMO_UNLOCK , 那么仅允许执行以下操作:
 - MQGMO_NO_SYNCPOINT
 - MQGMO_NO_WAIT

MQCLOSE 调用

对于 MQCLOSE 调用的选项:

- 不允许 MQCO_DELETE 和 MQCO_DELETE_PURGE 的组合。
- 仅允许下列其中一项:
 - MQCO_KEEP_SUB
 - MQCO_REMOVE_SUB

MQSUB 调用

对于 MQSUB 调用的选项:

- 必须至少指定下列其中一项:
 - MQSO_ALTER
 - MQSO_RESUME
 - MQSO_CREATE
- 仅允许下列其中一项:
 - MQSO_持久
 - MQSO_non_持久

注: 上面列出的选项是互斥的。但是, 由于 MQSO_NON_持久性的值为零, 因此将其与 MQSO_持久一起指定不会导致原因码 MQRC_OPTIONS_ERROR。提供了 MQSO_NON_持久性以帮助程序文档。

- 不允许 MQSO_GROUP_SUB 和 MQSO_MANAGED 的组合。
- MQSO_group_sub 需要指定 MQSO_SET_CORREL_ID。
- 仅允许下列其中一项:

- MQSO_ANY_USERID
- MQSO_FIXED_USERID
- 仅允许将 MQSO_NEW_publicATIONS_ONLY 与 MQSO_CREATE 结合使用。
- 不允许 MQSO_PUBLICATIONS_ON_REQUEST 与大于 1 的 SubLevel 的组合。
- 仅允许下列其中一项:
 - MQSO_WILDCARD_CHAR
 - MQSO_WILDCARD_TOPIC
- MQSO_NO_多点广播需要指定 MQSO_MANAGED。

已排队的发布/预订命令消息

应用程序可以使用 MQRFH2 命令消息来控制排队的发布/预订应用程序。

使用 MQRFH2 进行发布/预订的应用程序可以将以下命令消息发送到 SYSTEM.BROKER.CONTROL.QUEUE:

- [第 768 页的『删除发布消息』](#)
- [第 769 页的『注销订户消息』](#)
- [第 772 页的『发布消息』](#)
- [第 774 页的『注册订户消息』](#)
- [第 778 页的『请求更新消息』](#)

如果您正在编写排队的发布/预订应用程序，那么必须了解这些消息，队列管理器响应消息和消息描述符 (MQMD); 请参阅以下信息:

- [第 780 页的『队列管理器响应消息』](#)
- [第 784 页的『队列管理器转发的发布的 MQMD 设置』](#)
- [第 785 页的『队列管理器响应消息中的 MQMD 设置』](#)
- [第 781 页的『发布/预订原因码』](#)

这些命令包含在 MQRFH2 头的 **NameValueData** 字段中的 <psc> 文件夹中。可以由代理发送以响应命令消息的消息包含在 <pscr> 文件夹中。

每条命令的描述列出了文件夹内可以包含的属性。除非另行指定，否则这些属性是可选的，并且只能出现一次。

属性名称显示为 <Command>。

值必须为字符串格式，例如: Publish。

表示属性值的字符串常量显示在括号中，例如: (MQPSC_PUBLISH)。

字符串常量在队列管理器随附的头文件 cmqpsc.h 中定义。

删除发布消息

Delete Publication 命令消息将从发布程序或其他队列管理器发送到队列管理器，以指示队列管理器删除指定主题的任何保留发布。

此消息将发送到队列管理器的已排队发布/预订接口所监视的队列。

输入队列应当是原始发布内容被发送到的队列。

如果您对 **Delete Publication** 命令消息中指定的某些主题 (但不是所有主题) 具有权限，那么将仅删除这些主题。**Broker Response** 消息指示未删除哪些主题。

同理，如果 **Publish** 命令包含多个主题，**Delete Publication** 命令与其中一些相匹配 (非全部)，则仅删除 **Delete Publication** 命令中指定主题的发布内容。

请参阅第 784 页的『队列管理器转发的发布的 MQMD 设置』，以获取向队列管理器发送命令消息时所需的消息描述符 (MQMD) 参数的详细信息。

属性

<Command> (MQPSC_COMMAND)

该值为 DeletePub (MQPSC_DELETE_PUBLICATION)。
必须指定此属性。

<Topic> (MQPSC_TOPIC)

该值是包含要被删除的保留的发布内容的主题的字符串。字符串中可包含通配符以删除多个主题的发布内容。
必须指定此属性；可以为所需任意多的主题重复它。

<DelOpt> (MQPSC_DELETE_OPTION)

删除选项属性可以采用以下任一值：

Local (MQPSC_LOCAL)

将在本地队列管理器 (即，向其发送此消息的队列管理器) 上删除指定主题的所有保留发布内容，无论这些发布内容是否使用 **本地** 选项发布。

其他队列管理器上的发布不受影响。

None (MQPSC_NONE)

所有选项采取它们的缺省值。这与省略 **删除选项** 属性的效果相同。如果同时指定其他选项，则忽略 None。

如果省略此属性，那么缺省情况是在网络中的所有队列管理器上删除指定主题的所有保留发布，而不考虑是否使用 **本地** 选项发布这些发布。

示例

以下是 **Delete Publication** 命令消息的 NameValueData 示例。这由样本应用程序用于在本地队列管理器上删除包含 Team1 与 Team2 之间的匹配项中的最新分数的保留发布。

```
<psc>
  <Command>DeletePub</Command>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
  <DelOpt>Local</DelOpt>
</psc>
```

注销订户消息

Deregister Subscriber 命令消息由订户或另一个应用程序 (代表订户) 发送到队列管理器，以指示它不再希望接收与给定参数匹配的消息。

此消息将发送到 SYSTEM.BROKER.CONTROL.QUEUE，队列管理器的控制队列。用户必须具有将消息放入此队列所需的权限。

请参阅 **队列管理器转发的发布的 MQMD 设置**，以获取向队列管理器发送命令消息时所需的消息描述符 (MQMD) 参数的详细信息。

可以通过指定原始预订的相应主题，预订点和过滤器值来注销单个预订。如果未在原始预订中指定任何值 (即，它们使用缺省值)，那么在注销预订时应该省略这些值。

可以使用 **DeregAll** 选项来注销订户或一组订户的所有预订。例如，如果指定了 **DeregAll** 以及预订点 (但没有主题或过滤器)，那么将注销指定预订点上的订户的所有预订，而不考虑主题和过滤器。允许主题，过滤器和预订点的任意组合；如果全部三个都指定了一个预订，那么只能匹配一个预订，并且将忽略 **DeregAll** 选项。

消息必须由注册预订的订户发送；通过检查订户的用户标识来确认此消息。

系统管理员也可以使用 **MQSC** 或 **PCF** 命令注销预订。但是，向临时动态队列注册的预订与队列相关联，而不仅仅是队列名称。如果已显式删除队列，或者应用程序正在与队列管理器断开连接，那么无法再使用

Deregister Subscriber 命令来注销该队列的预订。可以使用开发者工作台来注销预订，并且队列管理器会在下次将发布与预订匹配时或下次队列管理器重新启动时自动除去这些预订。在正常情况下，应用程序应在删除队列或与队列管理器断开连接之前注销其预订。

如果订户发送消息以注销预订，并且接收到响应消息以表示已成功处理此消息，那么如果在注销预订的同时队列管理器正在处理某些发布内容，那么这些发布内容可能仍会到达订户队列。如果未从队列中除去消息，那么可能会在订户队列上堆积未处理的消息。如果应用程序在休眠一段时间后执行包含带有相应 `CorrelId` 的 `MQGET` 调用的循环，那么将从队列中清除这些消息。

同样，如果订户使用永久动态队列，并在 `MQCLOSE` 调用上使用 `MQCO_DELETE_PURGE` 选项注销和关闭该队列，那么该队列可能不为空。如果在删除队列时尚未落实队列管理器中的任何发布，那么 `MQCLOSE` 调用将发出 `MQRC_Q_NOT_EMPTY` 返回码。应用程序可以通过休眠并不时重新发出 `MQCLOSE` 调用来避免此问题。

属性

<Command> (MQPSC_COMMAND)

值为 `DeregSub (MQPSC_DEREGISTER_SUBSCRIBER)`。

必须指定此属性。

<Topic> (MQPSC_TOPIC)

该值是包含要注销的主题的字符串。

如果要注销多个主题，那么可以选择性地重复此属性。如果在 <RegOpt> 中指定了 `DeregAll`，那么可以将其省略。

如果订户希望保留其他主题的预订，那么指定的主题可以是已注册的主题的子集。允许使用通配符，但包含通配符的主题字符串必须与 **Deregister Subscriber** 命令消息中指定的相应字符串完全匹配。

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

该值是一个字符串，用于指定要从中拆离预订的预订点。

此属性不得重复。如果指定了 <Topic>，或者在 <RegOpt> 中指定了 `DeregAll`，那么可以将其省略。如果省略此属性，那么会发生以下情况：

- 如果执行不指定 `DeregAll`，那么将从缺省预订点注销与 <Topic> 属性 (以及 <Filter> 属性 (如果存在)) 匹配的预订。
- 如果指定 `DeregAll`，那么将从所有预订点注销所有预订 (与 <Topic> 和 <Filter> 属性 (如果存在) 匹配)。

请注意，不能显式指定缺省预订点。因此，无法仅从此预订点注销所有预订；必须指定主题。

<SubIdentity> (MQPSC_SUBSCRIPTION_IDENTITY)

这是最大长度为 64 个字符的可变长度字符串。它用于表示对预订感兴趣的应用程序。队列管理器为每个预订维护一组订户身份。每个预订可以允许其身份集仅持有单个身份或数量不限的身份。

如果 `SubIdentity` 位于预订的身份集中，那么会将其从该集合中除去。如果身份集因此变为空，那么将从队列管理器中除去预订，除非将 `LeaveOnly` 指定为 `RegOpt` 属性的值。如果身份集仍包含其他身份，那么不会从队列管理器中除去预订，并且不会中断发布流。

如果指定了 `SubIdentity`，但 `SubIdentity` 不在预订的身份集中，那么 **Deregister Subscriber** 命令将失败，返回码为 `MQRC_CF_SUB_IDENTITY_ERROR`。

<Filter> (MQPSC_FILTER)

该值是一个字符串，用于指定要注销的过滤器。它必须与先前已注册的预订过滤器完全匹配，包括大小写和任何空格。

如果要注销多个过滤器，那么可以选择性地重复此属性。如果指定了 <Topic>，或者在 <RegOpt> 中指定了 `DeregAll`，那么可以将其省略。

如果订户希望为其他过滤器保留预订，那么指定的过滤器可以是已注册的过滤器的子集。

<RegOpt> (MQPSC_REGLE_OPTION)

注册选项属性可以采用以下值：

DeregAll

(MQPSC_DEREGISTER_ALL)

将注销为此订户注册的所有匹配预订。

如果指定 DeregAll:

- 可以省略 <Topic>, <SubPoint>和 <Filter>。
- 如果需要, 可以重复 <Topic> 和 <Filter>。
- 不得重复 <SubPoint>。

如果未指定 DeregAll:

- 必须指定 <Topic>, 如果需要, 可以重复。
- 可以省略 <SubPoint> 和 <Filter>。
- 不得重复 <SubPoint>。
- 如果需要, 可以重复 <Filter>。

如果主题和过滤器都重复, 那么将除去与两者的所有组合匹配的所有预订。例如, 指定三个主题和三个过滤器的 **Deregister Subscriber** 命令将尝试除去九个预订。

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

消息描述符 (MQMD) 中的 CorrelId (不得为零) 用于标识订户。它必须与原始预订中使用的 CorrelId 匹配。

FullResp

(MQPSC_FULL_RESPONSE)

指定 FullResp 时, 如果命令未失败, 那么将在响应消息中返回预订的所有属性。

在 **Deregister Subscriber** 命令中不允许指定 FullResp 时 DeregAll。也无法指定多个主题。在这两种情况下, 该命令都失败, 返回码为 *MQRCCF_REG_OPTIONS_ERROR*。

LeaveOnly

(MQPSC_LEAVE_ONLY)

当您使用预订的身份集中的 SubIdentity 指定此值时, 将从预订的身份集中除去 SubIdentity。不会从队列管理器中除去预订, 即使生成的身份集为空也是如此。如果 SubIdentity 值不在身份集中, 那么该命令将失败, 返回码为 *MQRCCF_SUB_IDENTITY_ERROR*。

如果指定了 LeaveOnly 而未指定 SubIdentity, 那么该命令将失败, 返回码为 *MQRCCF_REG_OPTIONS_ERROR*。

如果既未指定 LeaveOnly 也未指定 SubIdentity, 那么将除去该预订, 而不考虑该预订的身份集的内容。

None

(MQPSC_NONE)

所有选项采取它们的缺省值。这与省略注册选项属性的效果相同。如果同时指定其他选项, 则忽略 None。

VariableUser 标识

(MQPSC_VARIABLE_USER_ID)

指定时, 订户 (队列, 队列管理器和 correlid) 的标识不限于单个用户标识。这与将原始注册消息的用户标识与订户的身份相关联的队列管理器的现有行为不同, 从此将阻止任何其他用户使用该身份。如果新订户尝试使用同一标识, 那么将返回返回码 *MQRCCF_DUPLICATE_SUBSCRIPTION*。

任何用户都可以在具有适当权限时修改或注销预订, 从而避免现有检查用户标识必须与原始订户的用户标识相匹配。

要将此选项添加到现有预订, 该命令必须来自与原始预订本身相同的用户标识。

如果要注销的预订设置了 `VariableUserId`，那么必须在注销时设置此项以指示要注销的预订。否则，将使用 **Deregister Subscriber** 命令的用户标识来标识预订。如果提供了预订名称，那么将覆盖此标识以及其他订户标识。

如果省略此属性，那么缺省情况是未设置注册选项。

<QMgrName> (MQPSC_Q_MGR_NAME)

该值是订户队列的队列管理器名称。它必须与原始预订中使用的 `QMgrName` 匹配。

如果省略此属性，缺省是消息描述符 (MQMD) 中的 `ReplyToQMgr` 名称。如果生成的名称为空，那么缺省为队列管理器的名称。

<QName> (MQPSC_Q_NAME)

该值是订户队列的名称。它必须与原始预订中使用的 `QName` 匹配。

如果省略此属性，那么缺省值为消息描述符 (MQMD) 中的 `ReplyToQ` 名称，不得为空。

<SubName> (MQPSC_SUBSCRIPTION_NAME)

如果在 **Deregister Subscriber** 命令上指定 `SubName`，那么 `SubName` 值优先于除用户标识以外的所有其他标识字段，除非在预订本身上设置了 `VariableUserId`。如果未设置 `VariableUserId`，那么仅当命令消息的用户标识与预订的用户标识匹配时，**Deregister Subscriber** 命令才会成功，如果命令未失败，返回码为 `MQRCCF_DUPLICATE_IDENTITY`。

如果存在与此命令的传统身份匹配但没有 `SubName` 的预订，那么 **Deregister Subscriber** 命令将失败，返回码为 `MQRCCF_SUB_NAME_ERROR`。如果尝试使用与传统身份匹配但未指定 `SubName` 的命令消息来注销具有 `SubName` 的预订，那么该命令将成功。

<SubUserData> (MQPSC_SUBSCRIPTION_USER_DATA)

这是一个变长文本字符串。该值由具有预订的队列管理器存储，但不会影响向订户交付发布内容。可以通过使用新值重新注册到同一预订来变更该值。此属性用于应用程序。

如果存在 `SubUser` 数据，那么将在预订的 `Metatopic` 信息 (`MQCACF_REG_SUB_USER_DATA`) 中返回 `SubUser` 数据。

示例

以下是 **Deregister Subscriber** 命令消息的 `NameValueData` 示例。在此示例中，样本应用程序正在注销其对包含所有匹配项的最新分数的主题的预订。订户的身份 (包括 `CorrelId`) 取自 MQMD 中的缺省值。

```
<psc>
  <Command>DeregSub</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

发布消息

Publish 命令消息将放入队列，或者从队列管理器放入订户，以发布有关一个或多个指定主题的信息。

将消息放入队列的权限和发布有关指定主题的信息的权限是必需的。

如果用户有权发布某些主题 (但不是所有主题) 的信息，那么仅使用这些主题进行发布; 警告响应指示哪些主题未用于发布。

如果订户具有任何匹配的预订，那么队列管理器会将 **Publish** 消息转发到相应的 **Register Subscriber** 命令消息中定义的订户队列。

请参阅 队列管理器响应消息，以获取向队列管理器发送命令消息时所需的消息描述符 (MQMD) 参数的详细信息，并在队列管理器将发布转发给订户时使用。

除非是本地发布，否则队列管理器会将 **Publish** 消息转发到网络中具有匹配预订的其他队列管理器。

发布数据 (如果有的话) 包括在消息的主体中。可以在 `MQRFH2` 头的 `NameValueData` 字段中的 `<mcd>` 文件夹中描述数据。

属性

<Command> (MQPSC_COMMAND)

值为 Publish(MQPSC_PUBLISH)。

必须指定此属性。

<Topic> (MQPSC_TOPIC)

该值是包含分类该发布内容的主题的字符串。不允许通配符。

必须将主题添加到名称列表 SYSTEM.QPUBSUB.QUEUE.NAMELIST，请参阅 [添加流](#) 以获取有关如何完成此任务的指示信息。

必须指定此属性，并且可选地能为所需任意多的主题重复它。

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

进行内容发布的预订点。

在 WebSphere Event Broker V6 中，<SubPoint> 属性的值是处理发布的发布节点的 "预订点" 属性的值。

在 WebSphere MQ V7.0.1 中，<SubPoint> 属性的值必须与预订点的名称匹配。请参阅 [添加预订点](#)。

<PubOpt> (MQPSC_PUBLICICATION_OPTION)

发布选项属性可以采用以下值：

RetainPub

(MQPSC_RETAIN_PUB)

队列管理器将保留发布的副本。如果未设置此选项，那么在队列管理器将发布内容发送至其所有当前订户后，将立即删除该发布内容。

IsRetainedPub

(MQPSC_IS_RETAINED_PUB)

(只能由队列管理器设置。) 此发布已由队列管理器保留。队列管理器设置此选项以通知订户此发布已提前发布并且已保留，前提是预订已向 InformIfRe 资格选项注册。它仅在响应 Register Subscriber 或 Request Update 命令消息时设置。直接被发送到订户的保留的发布内容没有此选项设置。

Local

(MQPSC_LOCAL)

此选项告知队列管理器，不得将此发布发送至其他队列管理器。在此队列管理器上注册的所有订户都将收到此发布内容 (如果他们具有匹配的预订)。

OtherSubsOnly

(MQPSC_OTHER_SUBS_ONLY)

此选项允许引用类型应用程序的较简单的处理，其中发布者也是对同一主题的订户。它指示队列管理器即使具有匹配的预订，也不要将发布发送到发布者的订户队列。发布者的订户队列由其 QMgrName, QName 和可选的 CorrelId 组成，如以下列表中所述。

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

MQMD (不能为零) 中的 CorrelId 是发布者的订户队列的一部分，在应用程序中发布者也是订户。

None

(MQPSC_NONE)

所有选项采取它们的缺省值。这与省略发布内容选项属性的效果相同。如果同时指定其他选项，则忽略 None。

您可以通过引入其他 <PubOpt> 元素来拥有多个发布选项。

如果属性被省略，则缺省是不设置发布选项。

<PubTime> (MQPSC_PUBLISH_TIMESTAMP)

该值是由发布者设置的可选的发布时间戳记。其格式为 16 个字符长度:

```
YYYYMMDDHHMSSSTH
```

使用通用时间。在发送到订户之前，队列管理器不会检查此信息。

<SeqNum> (MQPSC_SEQUENCE_NUMBER)

该值是由发布者设置的可选序列号。

它必须随每个发布一起按 1 递增。但是，队列管理器不会检查此信息，仅将此信息传输给订户。

如果将同一主题上的发布发布到不同的互连队列管理器，那么发布程序负责确保序号 (如果使用) 有意义。

<QMgrName> (MQPSC_Q_MGR_NAME)

该值是一个字符串，包含发布者的订户队列的队列管理器名称，在应用程序中发布者也是订户 (请参阅 OtherSubsOnly)。

如果省略此属性，缺省是消息描述符 (MQMD) 中的 ReplyToQMGr 名称。如果生成的名称为空，那么缺省为队列管理器的名称。

<QName> (MQPSC_Q_NAME)

该值是包含发布者的订户队列名称的字符串，在应用程序中发布者也是订户 (请参阅 OtherSubsOnly)。

如果省略此属性，缺省是消息描述符 (MQMD) 中的 ReplyToQ 名称，如果设置了 OtherSubsOnly 其不能为空。

示例

此处有一些 **Publish** 命令的 *NameValueData* 示例。

第一个示例用于，在示例应用程序中由匹配模拟器发送的发布内容，以表明已经启动匹配。

```
<psc>
  <Command>Publish</Command>
  <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

第二个示例用于保留的发布内容。发布了 Team1 和 Team2 之间比赛的最新比分。

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

注册订户消息

Register Subscriber 命令消息由订户或由代表订户的其他应用程序发送到队列管理器，以指示它要在预订点预订一个或多个主题。还可以指定消息内容过滤器。

在发布/预订过滤器表达式中，嵌套括号会导致性能成倍下降。避免将括号嵌套到大于约 6 的深度。

消息将发送到 SYSTEM.BROKER.CONTROL.QUEUE，这是队列管理器的控制队列。除了对预订中的一个或多个主题的访问权限 (由队列管理器的系统管理员设置) 之外，还需要将消息放入此队列的权限。

如果用户对某些 (但不是所有) 主题具有权限，那么仅注册具有权限的主题; 警告响应指示未注册的主题。

请参阅第 783 页的『到队列管理器的命令消息中的 MQMD 设置』，以获取向队列管理器发送命令消息时所需的消息描述符 (MQMD) 参数的详细信息。

如果应答队列是临时动态队列，那么队列管理器将在队列关闭时自动注销预订。

属性

<Command> (MQPSC_COMMAND)

值为 RegSub (MQPSC_REGISTER_SUBSCRIBER)。必须指定此属性。

<Topic> (MQPSC_TOPIC)

订户要接收其发布的主题。可以在主题中指定通配符。

如果使用 MQSC 命令 **display sub** 来检查以此方式创建的预订，那么 <Topic> 标记的值将显示为预订的 TOPICSTR 属性。

此属性是必需的，并且可以根据需要对任意数量的主题重复此属性。

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

该值是将预订附加到的预订点。

如果省略此属性，那么将使用缺省预订点。

在 WebSphere Event Broker V6 中，<SubPoint> 属性的值必须与预订的 Publication 节点的 Subscription Point 属性的值匹配。

在 WebSphere MQ V7.0.1 中，<SubPoint> 属性的值必须与预订点的名称匹配。请参阅 [添加预订点](#)。

<Filter> (MQPSC_FILTER)

该值是用作对发布消息内容的过滤器的 SQL 表达式。如果指定主题上的发布与过滤器匹配，那么会将其发送给订户。此属性对应于 MQSUB 和 MQOPEN 调用中使用的选择字符串。有关更多信息，请参阅 [选择消息内容](#)

如果省略此属性，那么不会进行内容过滤。

<RegOpt> (MQPSC_REGLE_OPTION)

此 "注册选项" 属性可以采用以下值:

AddName

(MQPSC_ADD_NAME)

为与此 "注册预订" 命令的传统身份相匹配的现有预订指定时，如果没有当前 SubName 值，那么会将此命令中指定的 SubName 添加到预订中。

如果指定了 AddName，那么 SubName 字段必填，否则将返回 MQRCCF_REG_OPTIONS_ERROR。

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

将匹配的发布发送到订户队列时，将使用消息描述符 (MQMD) 中的 CorrelId。CorrelId 不得为零，

FullResp

(MQPSC_FULL_RESPONSE)

指定时，如果命令未失败，那么将在响应消息中返回预订的所有属性。

仅当命令消息引用单个预订时，FullResp 才有效。因此，该命令中只允许一个主题；否则，该命令将失败，返回码为 MQRCCF_REG_OPTIONS_ERROR。

InformIfRet

(MQPSC_INFORM_IF_挽留)

队列管理器在发送发布消息以响应 **Register Subscriber** 或 **Request Update** 命令消息时通知订户是否保留发布内容。队列管理器通过在消息中包含 IsRetained 发布 发布选项来执行此操作。

JoinExcl

(MQPSC_JOIN_EXCLUSIVE)

此选项指示应该将指定的 SubIdentity 添加为预订的身份集的互斥成员，并且不能将其他身份添加到该集。

如果身份已加入 "共享" 并且是集合中的唯一条目, 那么该集合将更改为此身份持有的互斥锁定。否则, 如果预订当前在身份集中具有其他身份 (具有共享访问权), 那么该命令将失败, 返回码为 `MQRCCF_SUBSCRIPTION_IN_USE`。

JoinShared

(`MQPSC_JOIN_SHARED`)

此选项指示应该将指定的 `SubIdentity` 添加到预订的身份集。

如果当前以独占方式锁定了预订 (使用 `JoinExcl` 选项), 那么该命令将失败, 返回码为 `MQRCCF_SUBSCRIPTION_LOCKED`, 除非锁定了预订的身份与此命令消息中的身份相同。在这种情况下, 会自动将锁定修改为共享锁定。

Local

(`MQPSC_LOCAL`)

预订是本地预订, 不会分发给网络中的其他队列管理器。在其他队列管理器上进行的发布不会传递给此订户, 除非它还具有相应的全局预订。

NewPubs

(`MQPSC_NEW_PUBS_ONLY`)

注册预订时存在的保留发布不会发送给订户; 仅发送新发布。

如果订户重新注册并更改此选项以使其不再设置, 那么可能会再次发送已发送到它的发布。

NoAlter

(`MQPSC_NO_ALTER`)

不会更改现有匹配预订的属性。

创建预订时, 将忽略此选项。指定的所有其他选项都适用于新预订。

如果 `SubIdentity` 还指定了其中一个连接选项 (`JoinExcl` 或 `JoinShared`), 那么无论是否指定了 `NoAlter`, 都会将身份添加到身份集。

None

(`MQPSC_NONE`)

所有注册选项都采用其缺省值。

如果订户已注册, 那么其选项将重置为其缺省值 (请注意, 这不会与省略注册选项属性具有相同的影响), 并且将从 **Register Subscriber** 消息的 `MQMD` 更新预订到期时间。

如果同时指定了其他注册选项, 那么将忽略无。

NonPers

(`MQPSC_NON_PERSISTENT`)

与此预订匹配的发布将作为非持久消息传递给订户。

工作程序

(`MQPSC_PERSISTENT`)

与此预订匹配的发布将作为持久消息传递给订户。

PersAs 发布

(`MQPSC_PERSISTENT_AS_PUBLISH`)

与此预订匹配的发布将以发布者指定的持久性传递给订户。这是缺省行为。

PersAs 队列

(`MQPSC_PERSISTENT_AS_Q`)

与此预订匹配的发布将通过在订户队列上指定的持久性传递给订户。

PubOnReqOnly

(`MQPSC_PUB_ON_REQUEST_ONLY`)

除非响应 **Request Update** 命令消息, 否则队列管理器不会向订户发送发布内容。

VariableUser 标识

(MQPSC_VARIABLE_USER_ID)

指定时，订户 (队列，队列管理器和 correlid) 的标识不限于单个用户标识。这与将原始注册消息的用户标识与订户的身份相关联的队列管理器的现有行为不同，从此将阻止任何其他用户使用该身份。如果新订户尝试使用相同的标识，那么将返回 *MQRCCF_DUPLICATE_SUBSCRIPTION*。

这允许任何用户修改或注销预订 (如果该用户具有适当的权限)。因此，无需检查用户标识是否与原始订户的用户标识匹配。

要将此选项添加到现有预订，该命令必须来自与原始预订本身相同的用户标识。

如果 **Request Update** 命令的预订设置了 VariableUserId，那么必须在请求更新时设置此项以指示引用了哪个预订。否则，将使用 **Request Update** 命令的用户标识来标识预订。如果提供了预订名称，那么将覆盖此标识以及其他订户标识。

如果没有此选项集的 **Register Subscriber** 命令消息引用具有此选项集的现有预订，那么该选项将从此预订中除去，并且该预订的用户标识现在已固定。如果已存在具有相同身份 (队列，队列管理器和相关标识) 但与其关联的用户标识不同的订户，那么该命令将失败，返回码为 *MQRCCF_DUPLICATE_IDENTITY*，因为只能有一个用户标识与订户身份关联。

如果省略了注册选项属性，并且订户已注册，那么不会更改其注册选项，并且会从 **Register Subscriber** 消息的 MQMD 更新预订到期时间。

如果订户尚未注册，那么将创建一个新预订，所有注册选项都采用它们的缺省值。

缺省值为 PersAsPub，并且未设置任何其他选项。

<QMgrName> (MQPSC_Q_MGR_NAME)

该值是订户队列的队列管理器的名称，队列管理器将向该队列管理器发送匹配的发布。

如果省略此属性，缺省是消息描述符 (MQMD) 中的 ReplyToQMgr 名称。如果生成的名称为空，那么缺省为队列管理器的 QMgrName。

<QName> (MQPSC_Q_NAME)

该值是队列管理器向其发送匹配发布的订户队列的名称。

如果省略此属性，那么缺省值为消息描述符 (MQMD) 中的 ReplyToQ 名称，在这种情况下不得为空。

如果队列是临时动态队列，那么必须在 <RegOpt> 属性中指定发布 (NonPers) 的非持久传递。

如果队列是临时动态队列，那么队列管理器将在队列关闭时自动注销预订。

<SubName> (MQPSC_SUBSCRIPTION_NAME)

这是提供给特定预订的名称。您可以使用它来代替队列管理器，队列和可选的 correlId 来引用预订。

如果已存在具有此 **SubName** 的预订，那么该预订的任何其他属性 (Topic, QMgrName, QName, CorrelId, UserId, RegOpts, UserSubData 和 Expiry) 将被新的 Register Subscriber 命令消息中传递的属性 (如果已指定) 覆盖。但是，如果在未指定 QName 字段的情况下使用 **SubName**，并且在 MQMD 头中指定了 ReplyToQ，那么订户队列将更改为 ReplyToQ。

如果与此命令的传统身份匹配的预订已存在，但没有 **SubName**，那么除非指定了 **AddName** 选项，否则注册命令将失败，返回码为 *MQRCCF_DUPLICATE_SUBSCRIPTION*。

如果尝试使用另一个指定了相同 **SubName** 的 Register Subscriber 命令来更改现有指定的预订，并且新命令中的 Topic, QMgrName, QName 和 CorrelId 的值与另一个现有预订匹配 (无论是否定义了 SubName)，那么该命令将失败，返回码为 *MQRCCF_DUPLICATE_SUBSCRIPTION*。这将阻止两个预订名称引用同一预订。

<SubIdentity> (MQPSC_SUBSCRIPTION_IDENTITY)

此字符串用于表示对预订感兴趣的应用程序。它是最大长度为 64 个字符的可变长度字符串，并且是可选的。队列管理器为每个预订维护一组订户身份。每个预订可以允许其身份集仅包含一个身份或数量不限的身份 (请参阅 **JoinShared** 和 **JoinExcl** 选项)。

用于指定 **JoinShared** 或 **JoinExcl** 选项的预订命令将 **SubIdentity** 添加到预订的身份集 (如果该身份集尚未存在并且现有身份集允许执行此类操作); 即，没有其他订户以独占方式连接，或者该身份集为空。

由于 Register Subscriber 命令 (其中指定了 **SubIdentity**) 而对预订属性进行的任何更改, 仅当它是此预订的唯一身份集成员时才会成功。否则, 该命令将失败, 返回码为 *MQRCCF_SUBSCRIPTION_IN_USE*。这将防止预订的属性在其他感兴趣的订户不知晓的情况下发生更改。

如果指定长度超过 64 个字符的字符串, 那么该命令将失败, 返回码为 *MQRCCF_SUB_IDENTITY_ERROR*。

<SubUserData> (MQPSC_SUBSCRIPTION_USER_DATA)

这是一个变长文本字符串。该值由具有预订的队列管理器存储, 但对向订户的发布传递没有影响。可以通过使用新值重新注册到同一预订来变更该值。此属性可供应用程序使用。

如果存在预订的 Metatopic 信息 (*MQRCCF_REG_SUB_USER_DATA*) 中返回了 **SubUserData**。

如果指定多个注册选项值 NonPers, PersAsPub, PersAsQueue, and Pers, 那么仅使用最后一个值。不能在单个预订中组合这些选项。

示例

以下是 **Register Subscriber** 命令消息的 NameValueData 示例。在样本应用程序中, 结果服务使用此消息在设置了 "持久作为发布" 选项的情况下, 注册包含所有匹配项中最新分数的主题的预订。订户的身份 (包括 CorrelId) 取自 MQMD 中的缺省值。

```
<psc>
  <Command>RegSub</Command>
  <RegOpt>PersAsPub</RegOpt>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

请求更新消息

Request Update 命令消息从订户发送到队列管理器, 以请求与给定 (可选) 过滤器匹配的指定主题和预订点的当前保留发布。

此消息将发送到 *SYSTEM.BROKER.CONTROL.QUEUE*, 队列管理器的控制队列。除了请求更新中的主题访问权限外, 还需要将消息放入此队列的权限; 此权限由队列管理器的系统管理员设置。

如果订户在注册时指定了选项 PubOnReqOnly, 那么通常使用此命令。如果队列管理器具有任何匹配的保留发布, 那么会将这些发布发送给订户。如果队列管理器没有匹配的保留发布, 那么请求将失败, 返回码为 *MQRCCF_NO_RETAINED_MSG*。请求者必须先前已注册具有相同主题, SubPoint 和过滤器值的预订。

属性

<Command> (MQPSC_COMMAND)

值为 ReqUpdate (*MQPSC_REQUEST_UPDATE*)。必须指定此属性。

<Topic> (MQPSC_TOPIC)

该值是订户正在请求的主题; 允许使用通配符。

必须指定此属性, 但此消息中只允许出现一次。

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

该值是将预订附加到的预订点。

如果省略此属性, 那么将使用缺省预订点。

<Filter> (MQPSC_FILTER)

该值是用作对发布消息内容的过滤器的 ESQL 表达式。如果指定主题上的发布与过滤器匹配, 那么会将其发送给订户。

<Filter> 属性的值应该与您现在请求更新的原始预订上指定的值相同。

如果省略此属性, 那么不会进行内容过滤。

<RegOpt> (MQPSC_REGLE_OPTION)

注册选项属性可以采用以下值:

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

将匹配的发布发送到订户队列时，将使用消息描述符 (MQMD) 中的 CorrelId (不能为零)。

None

(MQPSC_NONE)

所有选项采取它们的缺省值。这与省略 <RegOpt> 属性的效果相同。如果同时指定其他选项，则忽略 None。

VariableUser 标识

(MQPSC_VARIABLE_USER_ID)

指定时，订户 (队列，队列管理器和 correlid) 的标识不限于单个用户标识。这与将原始注册消息的用户标识与订户的身份相关联的队列管理器的现有行为不同，从此将阻止任何其他用户使用该身份。如果新订户尝试使用同一身份，那么该命令将失败，返回码为 MQRCCF_DUPLICATE_SUBSCRIPTION。

这允许任何用户在具有适当限时修改或注销预订。因此，无需检查用户标识是否与原始订户的用户标识匹配。

要将此选项添加到现有预订，命令必须来自与原始预订相同的用户标识。

如果 **Request Update** 命令的预订设置了 VariableUserId，那么必须在请求更新时设置此项以指示引用了哪个预订。否则，将使用 **Request Update** 命令的用户标识来标识预订。如果提供了预订名称，那么将覆盖此标识以及其他订户标识。

如果省略此属性，那么缺省情况是未设置注册选项。

<QMgrName> (MQPSC_Q_MGR_NAME)

该值是订户队列的队列管理器的名称，队列管理器会将匹配的保留发布发送至该队列管理器。

如果省略此属性，缺省是消息描述符 (MQMD) 中的 ReplyToQMgr 名称。如果生成的名称为空，那么缺省为队列管理器的 QMgrName。

<QName> (MQPSC_Q_NAME)

该值是队列管理器将匹配的保留发布发送到的订户队列的名称。

如果省略此属性，那么缺省值为消息描述符 (MQMD) 中的 ReplyToQ 名称，在这种情况下不得为空。

<SubName> (MQPSC_SUBSCRIPTION_NAME)

这是提供给特定预订的名称。如果在 **Request Update** 命令上指定，那么 SubName 值优先于除用户标识以外的所有其他标识字段，除非在预订本身上设置了 VariableUserId。如果未设置 VariableUserId，那么仅当命令消息的用户标识与预订的用户标识匹配时，**Request Update** 命令才会成功。如果命令消息的用户标识与预订的用户标识不匹配，那么该命令将失败，返回码为 MQRCCF_DUPLICATE_IDENTITY。

如果设置了 VariableUserId，并且用户标识与预订的用户标识不同，那么如果新命令消息的用户标识有权浏览队列并将其放入预订的订户队列，那么该命令将成功。否则，该命令将失败，返回码为 MQRCCF_NOT_AUTHORIZED。

如果存在与此命令的传统身份相匹配的预订，但没有 SubName，那么 **Request Update** 命令将失败，返回码为 MQRCCF_SUB_NAME_ERROR。

如果尝试使用与传统身份匹配但未指定 SubName 的命令消息来请求更新具有 SubName 的预订，那么该命令将成功。

示例

以下是 **Request Update** 命令消息的 NameValueData 示例。在样本应用程序中，结果服务使用此消息来请求包含所有团队的最新评分的保留发布。订户的身份 (包括 CorrelId) 取自 MQMD 中的缺省值。

```
<psc>
  <Command>ReqUpdate</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

队列管理器响应消息

如果命令消息描述符指定需要响应，那么会将 **Queue Manager Response** 消息从队列管理器发送到发布者或订户的 ReplyToQ，以指示队列管理器接收到的命令消息是成功还是失败。

响应消息包含在 <pscr> 文件夹中 MQRFH2 头的 NameValueData 字段内。

在警告或错误的情况下，响应消息包含来自命令消息的 <psc> 文件夹以及 <pscr> 文件夹。消息数据 (如果有) 未包含在队列管理器响应消息中。在发生错误的情况下，导致错误的消息不会被处理，在发生警告的情况下，某些消息可能会被成功地处理。

如果有一个发送响应时发生错误：

- 对于发布消息，队列管理器尝试在 MQPUT 失败时将响应发送到 WebSphere MQ 死信队列。这允许将发布内容发送到发布者，即使响应无法被回送到发布者。
- 若出版响应无法发送到死信队列，错误将被登记入日志，命令消息通常会回滚。这种情况是否发生取决于 MQInput 节点是怎样定义的。

属性

<Completion> (MQPSCR_完成)

完成码可取三个值之一：

确定

命令成功完成

警告

完成但有警告。

错误

命令失败

<Response> (MQPSCR_RESPONSE)

对命令消息的响应发生在那条命令发出完成码为 warning 或 error 时。它包含 <Reason> 属性，并且可能包含指示警告或错误原因的其他属性。

在发生一个或多个错误的时候，只有一个响应文件夹表明第一个（也只有第一个）错误的原因。在发生一个或多个警告的时候，只有一个响应文件夹表明第一个（也只有第一个）警告的原因。

<Reason> (MQPSCR_REASON)

原因代码审核完成码，若 warning 或 error。它设置为以下示例中列出的其中一个错误代码。

<Reason> 属性包含在 <Response> 文件夹中。原因码可后跟 <psc> 文件夹中的任何有效属性（例如，主题名称），指示错误或警告的原因。如果您获得原因码 ???，check the data for correctness, for example, matching angled brackets (< >).

示例

以下是 **Queue Manager Response** 消息中 NameValueData 的一些示例。成功的响应可能如下：

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

此处有一个失败响应的示例：过滤器错误。第一个 NameValueData 字符串包含响应；第二个包含原始命令。

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
  ...
  command message (to which
  the queue manager is responding)
```

```
...
</psc>
```

此处有一个关于（由于未授权主题而产生）的警告响应。第一个 NameValueData 字符串包含响应; 第二个 NameValueData 字符串包含原始命令。

```
<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>
```

发布/预订原因码

这些原因码可能在发布/预订响应 <pscr> 文件夹的 **原因** 字段中返回。还列出了可用于在 C 或 C++ 编程语言中表示这些代码的常量。

MQRC_ 常量需要 WebSphere MQ cmqc.h 头文件。MQRCCF_ 常量需要 WebSphere MQ cmqcf.h 头文件 (除需要 cmqpsc.h 头文件的 MQRCCF_FILTER_ERROR 和 MQRCCF_WRONG_USER 外)。

| 原因码和文本 | 说明 | 发证方 |
|--------------------------------------|---|---------------|
| 2336 MQRC_RFH_COMMAND_ERROR | <pscr> 文件夹的 <Command> 字段的有效值为: RegSub, DeregSub, Publish, DeletePub 和 ReqUpdate。任何其他值都会导致发出此错误代码。 | 任何命令 |
| 2337 MQRC_RFH_PARM_ERROR | <pscr> 和 <mcd> 文件夹都有一组可以在其中指定的有效参数。请检查这些文件夹的描述, 并确保未指定不正确的参数。 | 任何命令 |
| 2338 MQRC_RFH_DUPLICATE_PARM | <pscr> 文件夹中的某些参数 (例如, 主题) 可以重复, 但其他参数 (例如, 命令) 不能重复。请检查您是否未复制不可重复的参数。 | 任何命令 |
| 2339 MQRC_RFH_PARM_MISSING | <pscr> 或 <mcd> 文件夹中的某些参数是可选的, 可以省略; 某些是必需的, 不得省略。检查是否在 <pscr> 和 <mcd> 文件夹中包含了所有必需参数。 | 任何命令 |
| 2551 MQRC_SELECTION_NOT_AVAILABLE | 没有扩展消息选择提供程序可用于确定具有指定过滤器的订户应该接收发布内容。 | 发布, 注册订户和请求更新 |
| | 没有扩展消息选择提供程序可用于处理指定订户的过滤器。 | 注册订户和请求更新 |
| 2554 MQRC_CONTENT_ERROR | 扩展消息选择提供程序在当前发布或保留发布中发现错误。 | 发布和请求更新 |

| 原因码和文本 | 说明 | 发证方 |
|----------------------------------|---|----------------------------------|
| 3008 MQRCCF_COMMAND_FAILED | 发生了阻止命令正确执行的内部错误。如果重新发出该命令，那么可能会发生此错误。队列管理器的系统事件日志包含向 IBM 报告问题时应使用的信息。 | 任何命令 |
| 3072 MQRCCF_TOPIC_ERROR | 您为 Topic 参数提供的一个或多个值不正确。检查主题的值是否符合指定的限制。 | 任何命令 |
| 3073 MQRCCF_NOT_REGISTERED | 在 DeregSub 或 ReqUpdate 命令上指定的 SubPoint, Topic 和 Filter 的组合不是先前已注册的组合, 对于 DeregSub 命令, 如果指定了 DeregAll 选项, 那么未使用其中一个 SubPoint, Topic 或 Filter 属性来注销任何预订。 | 注销订户和请求更新命令 |
| 3074 MQRCCF_Q_MGR_NAME_ERROR | 指定的队列管理器无效, 或者队列管理器不可用或不存在。 | "注销订户", "发布", "注册订户" 和 "请求更新" 命令 |
| 3076 MQRCCF_Q_NAME_ERROR | 指定的队列名称无效, 或者该队列在指定的队列管理器上不存在。 | "注销订户", "发布", "注册订户" 和 "请求更新" 命令 |
| 3077 MQRCCF_NO_RETAINED_MSG | 您指定的主题没有保留消息。这可能是错误, 也可能不是错误, 这取决于应用程序的设计。 | "请求更新" 命令 |
| 3079 MQRCCF_INCORRECT_Q | RegSub, DeregSub 和 ReqUpdate 命令始终发送到 SYSTEM.BROKER.CONTROL.QUEUE 队列, 这些队列将用于该队列管理器。将 "发布" 和 "删除发布" 命令发送到要用于它们的特定发布/预订消息流的输入队列; 这是在设计消息流时确定的。如果将命令发送到错误队列, 那么将返回此错误代码。 | 任何命令 |
| 3080 MQRCCF_CORREL_ID_ERROR | 您已指定 CorrelAs 标识作为其中一个 RegOpt 参数。但是, MQMD 的 CorrelId 字段不包含有效的相关标识 (即, 设置为 MQCI_NONE)。 | 注销订户和注册订户命令 |
| 3081 MQRCCF_NOT_AUTHORIZED | 未授权您执行所请求的操作。队列管理器的授权设置由系统管理员使用 "主题层次结构" 编辑器进行处理。 | 发布和注册订户命令 |
| 3083 MQRCCF_REG_OPTIONS_ERROR | 您在包含 RegSub 或 DeregSub 命令的 <psc> 文件夹中指定了无法识别的 RegOpt 参数。 | 注销订户和注册订户命令 |
| 3084 MQRCCF_PUB_OPTIONS_ERROR | 您在包含发布命令的 <psc> 文件夹中指定了无法识别的 PubOpt 参数。 | 发布命令 |
| 3087 MQRCCF_DEL_OPTIONS_ERROR | 您在包含 DeletePub 命令的 <psc> 文件夹中指定了无法识别的 DelOpt 参数。 | 删除发布命令 |

| 原因码和文本 | 说明 | 发证方 |
|---------------------------------------|--|------------------|
| 3150 MQRCCF_FILTER_ERROR | 为 "过滤器" 参数指定的值无效。请检查用于描述过滤器表达式的有效语法的部分，并确保表达式符合。 | 注销订户，注册订户和请求更新命令 |
| 3151 MQRCCF_WRONG_USER | 与指定的预订匹配的预订已存在;但是，它已由其他用户注册。预订只能由最初注册它的用户更改或注销。 | 注销订户，注册订户和请求更新命令 |
| 3152 MQRCCF_DUPLICATE_SUBSCRIPTION | 已存在具有其他预订名称的匹配预订。 | |
| 3153 MQRCCF_SUB_NAME_ERROR | 预订名称的格式无效，或者已存在没有预订名称的匹配预订。 | |
| 3154 MQRCCF_SUB_IDENTITY_ERROR | 预订身份参数错误。提供的值超过允许的最大长度，或者预订身份当前不是预订身份集的成员，并且未指定 "连接" 注册选项。 | |
| 3155 MQRCCF_SUBSCRIPTION_IN_USE | 当身份集的成员不是此集的唯一成员时，尝试修改或注销预订。 | |
| 3156 MQRCCF_SUBSCRIPTION_LOCKED | 预订当前被另一身份以独占方式锁定。 | |
| 3157 MQRCCF_ALREADY_JOINED | 已指定联合注册选项，但订户身份已是该预订的身份集的成员。 | |

到队列管理器的命令消息中的 MQMD 设置

将命令消息发送到队列管理器的应用程序使用消息描述符 (MQMD) 中的以下字段设置。此处未列出保留为缺省值的字段，或者可以按通常方式设置为任何有效值的字段。

Report

请参阅 `MsgType` 和 `CorrelId`。

MsgType

`MsgType` 应设置为 `MQMT_REQUEST` 或 `MQMT_DATAGRAM`。如果 `MsgType` 未设置为下列其中一个值，那么将返回 `MQRCCF_MSG_TYPE_ERROR`。

如果始终需要响应，那么应该将命令消息的 `MsgType` 设置为 `MQMT_REQUEST`。在这种情况下，`报告` 字段中的 `MQRO_PAN` 和 `MQRO_NAN` 标志不重要。

如果 `MsgType` 设置为 `MQMT_DATAGRAM`，那么响应取决于 `报告` 字段中 `MQRO_PAN` 和 `MQRO_NAN` 标志的设置：

- 仅 `MQRO_PAN` 表示仅当命令成功时，队列管理器才会发送响应。
- 仅 `MQRO_NAN` 表示仅当命令失败时，队列管理器才会发送响应。
- 如果命令完成并带有警告，那么如果设置了 `MQRO_PAN` 或 `MQRO_NAN`，那么将发送响应。
- `MQRO_PAN` + `MQRO_NAN` 表示无论命令成功还是失败，队列管理器都会发送响应。从队列管理器的角度来看，这与将 `MsgType` 设置为 `MQMT_REQUEST` 具有相同的效果。
- 如果既未设置 `MQRO_PAN` 也未设置 `MQRO_NAN`，那么不会发送任何响应。

FORMAT

设置为 `MQFMT_RF_HEADER_2`

MsgId

此字段通常设置为 MQMI_NONE，以便队列管理器生成唯一值。

CorrelId

此字段可以设置为任何值。如果发送方的身份包含 CorrelId，请在 **报告** 字段中指定此值以及 MQRO_PASS_CORREL_ID，以确保在队列管理器发送给发送方的所有响应消息中设置此值。

ReplyToQ

此字段定义要将响应 (如果有) 发送到的队列。这可能是发送方的队列; 它具有可以从消息中省略 QName 参数的优点。但是，如果要将响应发送到另一个队列，那么需要 QName 参数。

ReplyToQMgr

此字段定义响应的队列管理器。如果将此字段留空 (缺省值)，那么本地队列管理器将自己的名称放在此字段中。

队列管理器转发的发布的 MQMD 设置

队列管理器在向订户发送发布时使用消息描述符 (MQMD) 中的这些字段设置。MQMD 中的所有其他字段都设置为其缺省值。

Report

报告 设置为 MQRO_NONE。

MsgType

MsgType 设置为 MQMT_DATAGRAM。

到期

到期 设置为从发布程序接收到的 Publish 消息中的值。在保留消息的情况下，未完成的时间将减少消息在队列管理器中的大致时间。

FORMAT

格式 设置为 MQFMT_RF_HEADER_2

MsgId

MsgId 设置为唯一值。

CorrelId

如果 CorrelId 是订户身份的一部分，那么这是订户在注册时指定的值。否则，它是队列管理器选择的非零值。

优先级

优先级 采用发布程序设置的值，如果发布程序指定了 MQPRI_PRIORITY_AS_Q_DEF，那么将采用已解析的值。

持久性

持久性 采用发布程序设置的值，如果发布程序指定了 MQPER_PERSISTENCE_AS_Q_DEF，那么将采用此值，除非在发送此发布程序的订户的 Register Subscriber 消息中另有指定。

ReplyToQ

ReplyToQ 设置为空白。

ReplyToQMgr

ReplyToQMgr 设置为队列管理器的名称。

UserIdentifier

UserIdentifier 是订户的用户标识，如订户注册时所设置。

AccountingToken

AccountingToken 是订户的记帐令牌，如订户首次注册时所设置。

ApplIdentityData

ApplIdentity 数据 是订户的应用程序身份数据，如订户首次注册时所设置。

PutApplType

PutAppl 类型 设置为 MQAT_BROKER。

PutApplName

PutAppl 名称 设置为队列管理器名称的前 28 个字符。

PutDate

PutDate 是放入消息的日期。

PutTime

PutTime 是放入消息的时间。

ApplOriginData

ApplOrigin 数据 设置为空白。

队列管理器响应消息中的 MQMD 设置

队列管理器在发送对发布消息的应答时使用消息描述符 (MQMD) 中的这些字段设置。MQMD 中的所有其他字段都设置为其缺省值。

Report

报告 设置为全零。

MsgType

MsgType 设置为 MQMT_REPLY。

FORMAT

格式 设置为 MQFMT_RF_HEADER_2

MsgId

MsgId 的设置取决于原始命令消息中的 Report 选项。缺省情况下，它设置为 MQMI_NONE，以便队列管理器生成唯一值。

CorrelId

CorrelId 的设置取决于原始命令消息中的 报告 选项。缺省情况下，这意味着 CorrelId 设置为与命令消息的 MsgId 相同的值。这可用于使命令与其响应相关联。

优先级

优先级 设置为与原始命令消息中的值相同的值。

持久性

持久性 设置为原始命令消息中设置的值。

到期

到期 设置为与队列管理器接收到的原始命令消息中的值相同。

PutApplType

PutAppl 类型 设置为 MQAT_BROKER。

PutApplName

PutAppl 名称 设置为队列管理器名称的前 28 个字符。

其他上下文字段设置为如同使用 MQPMO_PASS_IDENTITY_CONTEXT 生成一样。

机器编码

本部分描述了消息描述符中 *Encoding* 字段的结构。

请参阅 [第 364 页的『MQMD - 消息描述符』](#) 以获取结构中字段的摘要。

Encoding 字段是一个 32 位整数，分为四个单独的子字段；这些子字段标识：

- 用于二进制整数的编码
- 用于压缩十进制整数的编码
- 用于浮点数的编码
- 保留位

每个子字段都由一个位掩码标识，该位掩码在对应于子字段的位置中具有 1 位，在其他位置具有 0 位。这些位的编号使得位 0 是最重要的位，而位 31 是最不重要的位。定义了以下掩码：

MQENC_INTEGER_MASK

二进制整数编码的掩码。

此子字段在 *Encoding* 字段中占据位位置 28 到 31。

MQENC_DECIMAL_MASK

packed-decimal-integer 编码的掩码。

此子字段在 *Encoding* 字段中占据位位置 24 到 27。

MQENC_FLOAT_MASK

浮点编码的掩码。

此子字段在 *Encoding* 字段中占据位位置 20 到 23。

MQENC_RESERVED_MASK

保留位的掩码。

此子字段在 *Encoding* 字段中占据位位置 0 到 19。

二进制整数编码

以下值对二进制整数编码有效:

MQENC_INTEGER_UNDEFINED

二进制整数使用未定义的编码表示。

MQENC_INTEGER_NORMAL

二进制整数以常规方式表示:

- 数字中最不重要的字节具有数字中任何字节的最高地址; 最重要的字节具有最低地址
- 每个字节中最不重要的位与具有下一个较高地址的字节相邻; 每个字节中最重要的位与具有下一个较低地址的字节相邻

MQENC_INTEGER_REVERSED

二进制整数的表示方式与 MQENC_INTEGER_NORMAL 相同, 但字节排列顺序相反。每个字节中的位的排列方式与 MQENC_INTEGER_NORMAL 相同。

压缩十进制整数编码

以下值对压缩十进制整数编码有效:

MQENC_DECIMAL_UNDEFINED

使用未定义的编码表示压缩十进制整数。

MQENC_DECIMAL_NORMAL

以传统方式表示压缩十进制整数:

- 数字的可打印格式中的每个十进制数字都以压缩十进制表示, 范围为 X'0'到 X'9' 的单个十六进制数字。每个十六进制数字占用四个位, 因此压缩十进制数字中的每个字节表示数字的可打印形式的两位十进制数字。
- 压缩十进制数中最不重要的字节是包含最不重要的十进制数字的字节。在该字节中, 最重要的四个位包含最不重要的十进制数字, 最不重要的四个位包含符号。符号为 X'C' (正), X'D' (负) 或 X'F' (无符号)。
- 数字中最不重要的字节具有数字中任何字节的最高地址; 最重要的字节具有最低地址。
- 每个字节中最不重要的位与具有下一个较高地址的字节相邻; 每个字节中最重要的位与具有下一个较低地址的字节相邻。

MQENC_DECIMAL_REVERSED

压缩十进制整数的表示方式与 MQENC_DECIMAL_NORMAL 相同, 但字节排列顺序相反。每个字节中的位的排列方式与 MQENC_DECIMAL_NORMAL 相同。

浮点编码

以下值对浮点编码有效:

MQENC_FLOAT_UNDEFINED

浮点数使用未定义的编码表示。

MQENC_FLOAT_IEEE_NORMAL

使用标准 IEEE 表示浮点数³ 浮点格式，字节排列如下：

- 尾数中最不重要的字节具有数字中任何字节的最高地址；包含指数的字节具有最低地址
- 每个字节中最不重要的位与具有下一个较高地址的字节相邻；每个字节中最重要的是与具有下一个较低地址的字节相邻

IEEE 浮点编码的详细信息可在 IEEE 标准 754 中找到。

MQENC_FLOAT_IEEE_REVERSED

浮点数的表示方式与 MQENC_FLOAT_IEEE_NORMAL 相同，但字节排列顺序相反。每个字节中的位的排列方式与 MQENC_FLOAT_IEEE_NORMAL 相同。

MQENC_FLOAT_S390

浮点数使用标准 System/390 浮点格式表示；这也由 System/370 使用。

构造编码

要在 MQMD 中构造 *Encoding* 字段的值，描述所需编码的相关常量可以是：

- 一起添加，或
- 使用按位 OR 运算进行组合 (如果编程语言支持位运算)

无论使用哪种方法，仅将一个 MQENC_INTEGER_* 编码与一个 MQENC_DECIMAL_* 编码和一个 MQENC_FLOAT_* 编码组合在一起。

分析编码

Encoding 字段包含子字段；因此，需要检查整数，压缩十进制或浮点编码的应用程序必须使用所描述的其中一种方法。

使用位操作

如果编程语言支持位操作，请执行以下步骤：

1. 根据所需的编码类型，选择下列其中一个值：
 - MQENC_INTEGER_MASK 用于二进制整数编码
 - 压缩十进制整数编码的 MQENC_DECIMAL_MASK
 - MQENC_FLOAT_MASK 用于浮点编码调用值 A。
2. 使用按位 AND 运算将 *Encoding* 字段与 A 组合在一起；调用结果 B。
3. B 是必需的编码，可以使用对该编码类型有效的每个值进行等式检验。

使用算术

如果编程语言不支持位操作，请使用整数算术执行以下步骤：

1. 根据所需的编码类型，选择下列其中一个值：
 - 1 表示二进制整数编码
 - 16 表示压缩十进制整数编码
 - 256 表示浮点编码调用值 A。
2. 将 *Encoding* 字段的值除以 A；调用结果 B。
3. 将 B 除以 16；调用结果 C。

³ 电气与电子工程师学院

4. 将 C 乘以 16 并从 B 中减去; 调用结果 D。
5. 将 D 乘以 A; 调用结果 E。
6. E 是必需的编码, 可以使用对该编码类型有效的每个值进行等式检验。

机器体系结构编码的摘要

第 788 页的表 578 中显示了机器体系结构的编码。

| 机器体系结构 | 二进制整数编码 | 压缩十进制整数编码 | 浮点编码 |
|------------|----------|-----------|------------|
| IBM i | 正态 | 正态 | IEEE 正常 |
| Intel x86 | reversed | reversed | IEEE 已撤销 |
| PowerPC | 正态 | 正态 | IEEE 正常 |
| System/390 | 正态 | 正态 | System/390 |

报告选项和消息标志

本部分描述了作为 MQGET, MQPUT 和 MQPUT1 调用上指定的消息描述符 MQMD 的一部分的 *Report* 和 *MsgFlags* 字段。

本节中的主题描述:

- 报告字段的结构以及队列管理器如何处理该报告字段
- 应用程序如何分析报告字段
- 消息标志字段的结构

有关 MQMD 消息描述符的更多信息, 请参阅第 364 页的『MQMD - 消息描述符』。

报告字段的结构

此信息描述报告字段的结构。

Report 字段是一个 32 位整数, 分为三个单独的子字段。这些子字段标识:

- 本地队列管理器无法识别时被拒绝的报告选项
- 报告始终接受的选项, 即使本地队列管理器无法识别这些选项也是如此
- 仅当满足某些其他条件时才接受的报告选项

每个子字段都由一个位掩码标识, 该位掩码在对应于子字段的位置中具有 1 位, 在其他位置具有 0 位。子字段中的位不一定相邻。这些位的编号使得位 0 是最重要的位, 而位 31 是最不重要的位。定义了以下掩码以标识子字段:

MQRO_REJECT_UNSUP_MASK

此掩码标识 *Report* 字段中的位位置, 本地队列管理器不支持的报告选项会导致 MQPUT 或 MQPUT1 调用失败, 完成代码为 MQCC_FAILED, 原因码为 MQRC_REPORT_OPTIONS_ERROR。

此子字段占用位位置 3 以及 11 到 13。

MQRO_ACCEPT_UNSUP_MASK

此掩码标识 *Report* 字段中的位位置, 但是在 MQPUT 或 MQPUT1 调用上接受本地队列管理器不支持的报告选项。在此情况下, 将返回完成代码 MQCC_WARNING, 原因码为 MQRC_UNKNOWN_REPORT_OPTION。

此子字段占用位位置 0 到 2, 4 到 10 以及 24 到 31。

以下报告选项包含在此子字段中:

- 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

此掩码标识 *Report* 字段中的位位置，在此字段中，在 MQPUT 或 MQPUT1 调用上仍然接受本地队列管理器不支持的报告选项，但前提是满足以下两个条件：

- 该消息的目标是远程队列管理器。
- 应用程序不会将消息直接放在本地传输队列上 (即，由 MQOPEN 或 MQPUT1 调用上指定的对象描述符中的 *ObjectQMgrName* 和 *ObjectName* 字段标识的队列不是本地传输队列)。

如果满足这些条件，那么将返回带有原因码 MQRC_UNKNOWN_REPORT_OPTION 的完成代码 MQCC_WARNING，如果未满足这些条件，那么将返回带有原因码 MQRC_REPORT_OPTIONS_ERROR 的 MQCC_FAILED。

此子字段占用位位置 14 到 23。

以下报告选项包含在此子字段中：

- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA

如果在队列管理器无法识别的 *Report* 字段中指定了任何选项，那么队列管理器会依次通过使用按位 AND 操作将 *Report* 字段与该子字段的掩码组合来检查每个子字段。如果该操作的结果不为零，那么将返回上述完成代码和原因码。

如果返回 MQCC_WARNING，那么未定义在存在其他警告条件时返回的原因码。

当使用由远程队列管理器识别和处理的报告选项发送消息时，指定并具有本地队列管理器无法识别的已接受报告选项的能力很有用。

分析报告字段

Report 字段包含子字段；因此，需要检查消息发送方是否请求了特定报告的应用程序必须使用所描述的其中一种方法。

使用位操作

如果编程语言支持位操作，请执行以下步骤：

1. 根据要检查的报告类型，选择下列其中一个值：

- 针对 COA 报告的 MQRO_COA_WITH_FULL_DATA
- 针对 COD 报告的 MQRO_COD_WITH_FULL_DATA
- 异常报告的 MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA 用于到期报告

调用值 A。

在 z/OS 上，使用 MQRO_*_WITH_DATA 值而不是 MQRO_*_WITH_FULL_DATA 值。

2. 使用按位 AND 运算将 *Report* 字段与 A 组合在一起; 调用结果 B。
3. 测试 B 是否与该类型的报告可能具有每个值相同。

例如，如果 A 是 MQRO_EXCEPTION_WITH_FULL_DATA，请测试 B 是否与以下各项相同，以确定由消息发送方指定的内容：

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

无论对应用程序逻辑最方便的顺序是什么，都可以执行这些测试。

使用类似的方法来测试 MQRO_PASS_MSG_ID 或 MQRO_PASS_CORREL_ID 选项；选择这两个常量中的任何一个作为值 A，然后按上述方式继续操作。

使用算术

如果编程语言不支持位操作，请使用整数算术执行以下步骤：

1. 根据要检查的报告类型，选择下列其中一个值：
 - MQRO_COA for COA 报告
 - MQRO_COD for COD 报告
 - 异常报告的 MQRO_EXCEPTION
 - 到期报告的 MQRO_EXPIRATION
 调用值 A。
2. 将 *Report* 字段除以 A；调用结果 B。
3. 将 B 除以 8；调用结果 C。
4. 将 C 乘以 8 并从 B 中减去；调用结果 D。
5. 将 D 乘以 A；调用结果 E。
6. 测试 E 是否与该类型的报告可能具有每个值相同。

例如，如果 A 是 MQRO_EXCEPTION，请使用以下各项测试 E 是否相等，以确定消息发送方指定的内容：

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

无论对应用程序逻辑最方便的顺序是什么，都可以执行这些测试。

以下伪代码说明了异常报告消息的此方法：

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

使用类似的方法来测试 MQRO_PASS_MSG_ID 或 MQRO_PASS_CORREL_ID 选项; 选择相应的值 A 作为这两个常量中的任何一个, 然后按上文所述继续操作, 但将以上步骤中的值 8 替换为值 2。

消息标志字段的结构

此信息描述了 message-flags 字段的结构。

MsgFlags 字段是一个 32 位整数, 分为三个单独的子字段。这些子字段标识:

- 本地队列管理器无法识别时拒绝的消息标志
- 始终接受的消息标志, 即使本地队列管理器无法识别它们
- 仅当满足某些其他条件时才接受的消息标志

注: *MsgFlags* 中的所有子字段都保留供队列管理器使用。

每个子字段都由一个位掩码标识, 该位掩码在对应于子字段的位置中具有 1 位, 在其他位置具有 0 位。这些位的编号使得位 0 是最重要的位, 而位 31 是最不重要的位。定义了以下掩码以标识子字段:

MQMF_REJECT_UNSUP_MASK

此掩码标识 *MsgFlags* 字段中的位位置, 其中本地队列管理器不支持的消息标志导致 MQPUT 或 MQPUT1 调用失败, 完成代码为 MQCC_FAILED, 原因码为 MQRC_MSG_FLAGS_ERROR。

此子字段占用位位置 20 到 31。

此子字段中包含以下消息标志:

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_ALLOWED

MQMF_ACCEPT_UNSUP_MASK

此掩码标识 *MsgFlags* 字段中的位位置, 在此字段中, 在 MQPUT 或 MQPUT1 调用上仍然接受本地队列管理器不支持的消息标志。完成代码为 MQCC_OK。

此子字段占用位位置 0 到 11。

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK

此掩码标识 *MsgFlags* 字段中的位位置, 在此字段中, 在 MQPUT 或 MQPUT1 调用上仍然接受本地队列管理器不支持的消息标志, 但前提是满足以下两个条件:

- 该消息的目标是远程队列管理器。
- 应用程序不会将消息直接放在本地传输队列上 (即, 由 MQOPEN 或 MQPUT1 调用上指定的对象描述符中的 *ObjectQMgrName* 和 *ObjectName* 字段标识的队列不是本地传输队列)。

如果满足这些条件, 那么将返回完成代码 MQCC_OK, 否则返回原因码为 MQRC_MSG_FLAGS_ERROR 的 MQCC_FAILED。

此子字段占用位位置 12 到 19。

如果在 *MsgFlags* 字段中指定了队列管理器无法识别的标志, 那么队列管理器将通过使用按位 AND 操作将 *MsgFlags* 字段与该字段的掩码组合来依次检查每个子字段。如果该操作的结果不为零, 那么将返回上述完成代码和原因码。

数据转换出口

此主题集合描述数据转换出口的接口, 以及队列管理器在需要数据转换时执行的处理。

有关数据转换的更多信息, 请参阅 *WebSphere MQ* 下的数据转换 (<https://www.ibm.com/support/docview.wss?uid=swg27005729>)。

在 MQGET 调用的处理过程中调用数据转换出口，以便将应用程序消息数据转换为接收应用程序所需的表示。应用程序消息数据的转换是可选的；它需要在 MQGET 调用上指定 MQGMO_CONVERT 选项。

描述了以下主题：

- 队列管理器为响应 MQGMO_CONVERT 选项而执行的处理；请参阅第 792 页的『转换处理』。
- 处理内置格式时队列管理器使用的处理约定；也建议用户编写的出口使用这些约定。请参阅第 793 页的『处理约定』。
- 转换报告消息的特殊注意事项；请参阅第 796 页的『报告消息的转换』。
- 传递到数据转换出口的参数；请参阅第 807 页的『MQ_DATA_CONV_EXIT-数据转换出口』。
- 可从出口用于在不同表示之间转换字符数据的调用；请参阅第 802 页的『MQXCNCV-转换字符』。
- 特定于出口的数据结构参数；请参阅第 797 页的『MQDXP-数据转换出口参数』。

转换处理

此信息描述队列管理器为响应 MQGMO_CONVERT 选项而执行的处理。

如果在 MQGET 调用上指定了 MQGMO_CONVERT 选项，并且存在要返回到应用程序的消息，那么队列管理器将执行以下操作：

1. 如果满足以下一项或多项条件，那么无需进行转换：
 - 消息数据已包含在发出 MQGET 调用的应用程序所需的字符集和编码中。在发出调用之前，应用程序必须将 MQGET 调用的 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段设置为必需值。
 - 消息数据的长度为零。
 - MQGET 调用的 *Buffer* 参数的长度为零。

在这些情况下，将返回消息而不转换为发出 MQGET 调用的应用程序；*MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 值将设置为消息中的控制信息中的值，并且调用将通过完成代码和原因的下列其中一个组合来完成：

| 完成代码 | 原因码 |
|--------------|-----------------------------|
| MQCC_OK | MQRC_NONE |
| MQCC_WARNING | MQRC_TRUNCATED_MSG_RECEIVED |
| MQCC_WARNING | MQRC_TRUNCATED_MSG_FAILED |

仅当消息数据的字符集或编码与 *MsgDesc* 参数中的相应值不同，并且存在要转换的数据时，才会执行以下步骤：

2. 如果消息中的控制信息中的 *Format* 字段具有值 MQFMT_NONE，那么将返回未转换的消息，完成代码为 MQCC_WARNING，原因码为 MQRC_FORMAT_ERROR。
在所有其他情况下，将继续进行转换处理。
3. 将从队列中除去该消息，并将其放入与 *Buffer* 参数大小相同的临时缓冲区中。对于浏览操作，消息将复制到临时缓冲区中，而不是从队列中除去。
4. 如果必须截断消息以适合缓冲区，那么将执行以下操作：
 - 如果未指定 MQGMO_ACCEPT_TRUNCATED_MSG 选项，那么将返回未转换的消息，完成代码为 MQCC_WARNING，原因码为 MQRC_TRUNCATED_MSG_FAILED。
 - 如果指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项，那么完成代码将设置为 MQCC_WARNING，原因码将设置为 MQRC_TRUNCATED_MSG_ACCEPT，并且转换处理将继续。
5. 如果可以在缓冲区中容纳消息而不截断，或者指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项，那么将执行以下操作：
 - 如果格式是内置格式，那么缓冲区将传递到队列管理器的数据转换服务。
 - 如果该格式不是内置格式，那么会将缓冲区传递到与该格式同名的用户编写的出口。如果找不到出口，那么将返回未转换的消息，完成代码为 MQCC_WARNING，原因码为 MQRC_FORMAT_ERROR。

如果未发生错误，那么来自数据转换服务或用户编写的出口的输出是已转换的消息，以及要返回到发出 MQGET 调用的应用程序的完成代码和原因码。

6. 如果转换成功，那么队列管理器会将转换后的消息返回给应用程序。在这种情况下，MQGET 调用返回的完成代码和原因码是下列其中一种组合：

| 完成代码 | 原因码 |
|--------------|-----------------------------|
| MQCC_OK | MQRC_NONE |
| MQCC_WARNING | MQRC_TRUNCATED_MSG_RECEIVED |

但是，如果转换由用户编写的出口执行，那么即使转换成功，也可以返回其他原因码。

如果转换失败，那么队列管理器会将未转换的消息返回到应用程序，并且 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段将设置为消息中控制信息中的值，并且完成代码为 MQCC_WARNING。

处理约定

转换内置格式时，队列管理器遵循描述的处理约定。

用户编写的出口也应遵循这些约定，尽管队列管理器不会强制这样做。队列管理器转换的内置格式包括：

- MQFMT_ADMIN
- MQFMT_CICS (仅限 z/OS)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT V 1
- MQFMT_EVENT V 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 (仅限 z/OS)
- MQFMT_XMIT_Q_HEADER

1. 如果消息在转换期间扩展，并且超出 *Buffer* 参数的大小，那么将执行以下操作：

- 如果未指定 MQGMO_ACCEPT_TRUNCATED_MSG 选项，那么将返回未转换的消息，完成代码为 MQCC_WARNING，原因码为 MQRC_CONVERTED_MSG_TOO_BIG。
- 如果指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项是，那么将截断消息，将完成代码设置为 MQCC_WARNING，将原因码设置为 MQRC_TRUNCATED_MSG_ACCEPT，然后继续进行转换处理。

2. 如果发生截断 (在转换之前或期间)，那么 *Buffer* 参数中返回的有效字节数可以小于缓冲区的长度。

例如，如果 4 字节整数或 DBCS 字符跨越缓冲区的末尾，那么会发生此情况。未转换信息的不完整元素，并且返回的消息中的那些字节不包含有效信息。如果在转换期间缩减之前截断的消息，也会发生此情况。

如果返回的有效字节数小于缓冲区的长度，那么缓冲区末尾未使用的字节将设置为空。

3. 如果数组或字符串跨越缓冲区的末尾，那么将转换尽可能多的数据；仅转换不完整的特定数组元素或 DBCS 字符；转换前面的数组元素或字符。
4. 如果发生截断（在转换之前或期间），那么针对 *DataLength* 参数返回的长度是截断之前未转换消息的长度。
5. 当在单字节字符集 (SBCS)，双字节字符集 (DBCS) 或多字节字符集 (MBCS) 之间转换字符串时，这些字符串可以扩展或收缩。

- 在 PCF 格式 MQFMT_ADMIN，MQFMT_EVENT 和 MQFMT_PCF 中，MQCFST 和 MQCFSL 结构中的字符串会根据需要进行扩展或收缩，以适应转换后的字符串。

对于字符串列表结构 MQCFSL，列表中的字符串可能会扩展或收缩不同的金额。如果发生这种情况，那么队列管理器将用空格填充较短的字符串，以使它们与转换后最长的字符串具有相同的长度。

- 在 MQFMT_REF_MSG_HEADER 格式中，*SrcEnvOffset*，*SrcNameOffset*，*DestEnvOffset* 和 *DestNameOffset* 字段所寻址的字符串会根据需要进行扩展或收缩，以在转换后容纳这些字符串。
- 在 MQFMT_RF_HEADER 格式中，*NameValueString* 字段会根据需要展开或收缩以适应转换后的名称/值对。
- 在具有固定字段大小的结构中，队列管理器允许字符串在其固定字段中展开或收缩，前提是不会丢失重要信息。在此方面，字段中第一个空字符之后的尾部空格和字符被视为无关紧要。
 - 如果字符串扩展，但仅需要废弃无关紧要的字符以在字段中容纳转换后的字符串，那么转换将成功，并且调用将完成，并带有 MQCC_OK 和原因码 MQRC_NONE (假定没有其他错误)。
 - 如果字符串扩展，但转换后的字符串需要废弃有效字符以适合该字段，那么将返回未转换的消息，并且调用将完成，并带有 MQCC_WARNING 和原因码 MQRC_CONVERTED_STRING_TOO_BIG。

注：无论是否指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项，原因码 MQRC_CONVERTED_STRING_TOO_BIG 都会导致此情况。

- 如果字符串收缩，那么队列管理器将用空白填充字符串以达到字段的长度。

6. 对于由一个或多个 MQ 头结构后跟用户数据组成的消息，可能会转换一个或多个头结构，而不会转换其余消息。但是，(有两个例外) 每个头结构中的 *CodedCharSetId* 和 *Encoding* 字段始终正确指示跟在头结构后面的数据的字符集和编码。

两个例外是 MQCIH 和 MQIIH 结构，其中这些结构中的 *CodedCharSetId* 和 *Encoding* 字段中的值不重要。对于这些结构，结构后面的数据与 MQCIH 或 MQIIH 结构本身的字符集和编码相同。

7. 如果要检索的消息的控制信息中的 *CodedCharSetId* 或 *Encoding* 字段，或者在 *MsgDesc* 参数中指定未定义或不受支持的值，那么如果在转换消息时不需要使用未定义或不受支持的值，那么队列管理器可能会忽略该错误。

例如，如果消息中的 *Encoding* 字段指定了不受支持的浮点编码，但消息仅包含整数数据，或者包含不需要转换的浮点数据 (因为源和目标浮点编码相同)，那么可能无法诊断错误。

如果诊断错误，那么将返回未转换的消息，完成代码为 MQCC_WARNING 和其中一个 MQRC_SOURCE_*_ERROR 或 MQRC_TARGET_*_ERROR 原因码 (视情况而定)；*MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段将设置为消息中控制信息中的值。

如果未诊断错误并且转换成功完成，那么在 *MsgDesc* 参数的 *CodedCharSetId* 和 *Encoding* 字段中返回的值是由发出 MQGET 调用的应用程序指定的值。

8. 在所有情况下，如果将消息返回到未转换的应用程序，那么完成代码将设置为 MQCC_WARNING，并且 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段将设置为适合于未转换的数据的值。这也是针对 MQFMT_NONE 完成的。

Reason 参数设置为指示无法执行转换的原因的代码，除非还必须截断消息；与截断相关的原因码优先于与转换相关的原因码。(要确定是否转换了截断的消息，请检查 *MsgDesc* 参数的 *CodedCharSetId* 和 *Encoding* 字段中返回的值。)

诊断错误时，将返回特定原因码，或者返回一般原因码 MQRC_NOT_CONVERTED。返回的原因码取决于底层数据转换服务的诊断功能。

9. 如果返回完成代码 MQCC_WARNING，并且多个原因码相关，那么优先顺序如下：

a. 以下原因优先于所有其他原因; 只能出现此组中的其中一个原因:

- MQRC_SIGNAL_REQUEST_接受
- MQRC_TRUNCATED_MSG_RECEIVED

b. 未定义其余原因码中的优先顺序。

10. MQGET 调用完成时:

- 以下原因码指示消息已成功转换:
 - MQRC_NONE
- 以下原因码指示消息可能已成功转换 (请检查 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段以查找):
 - MQRC_MSG_MARKED_BROWSE_CO_OP
 - MQRC_TRUNCATED_MSG_RECEIVED
- 所有其他原因码都指示未转换消息。

以下处理特定于内置格式; 它不适用于用户定义的格式:

11. 以下格式除外:

- MQFMT_ADMIN
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_EVENT
- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- MQFMT_STRING

对于队列名称中有效的字符, 不能将任何内置格式转换为不具有 SBCS 字符的字符集。如果尝试执行此类转换, 那么将返回未转换的消息, 完成代码为 MQCC_WARNING, 原因码为 MQRC_SOURCE_CCSID_ERROR 或 MQRC_TARGET_CCSID_ERROR (视情况而定)。

Unicode 字符集 UCS-2 是一个字符集的示例, 对于队列名称中有效的字符, 该字符集没有 SBCS 字符。

12. 如果截断内置格式的消息数据, 那么将不调整消息中包含字符串长度或元素或结构计数的字段, 以反映实际返回到应用程序的数据长度; 针对消息数据中的此类字段返回的值是适用于截断前消息的值。

处理消息 (例如, 截断的 MQFMT_ADMIN 消息) 时, 请确保应用程序不会尝试访问超出返回的数据末尾的数据。

13. 如果格式名称为 MQFMT_DEAD_LETTER_HEADER, 那么消息数据以 MQDLH 结构开头, 可能后跟零个或多个字节的应用程序消息数据。应用程序消息数据的格式, 字符集和编码由消息开头的 MQDLH 结构中的 *Format*, *CodedCharSetId* 和 *Encoding* 字段定义。由于 MQDLH 结构和应用程序消息数据可以具有不同的字符集和编码, 因此 MQDLH 结构和应用程序消息数据中的一个或两个可能需要转换。

根据需要, 队列管理器首先转换 MQDLH 结构。如果转换成功, 或者 MQDLH 结构不需要转换, 那么队列管理器会检查 MQDLH 结构中的 *CodedCharSetId* 和 *Encoding* 字段, 以查看是否需要转换应用程序消息数据。如果需要转换, 那么队列管理器将使用 MQDLH 结构中的 *Format* 字段给出的名称来调用用户编写的出口, 或者执行转换本身 (如果 *Format* 是内置格式的名称)。

如果 MQGET 调用返回完成代码 MQCC_WARNING, 并且原因码是指示转换不成功的原因码之一, 那么下列其中一项适用:

- 无法转换 MQDLH 结构。在这种情况下, 也不会转换应用程序消息数据。
- 已转换 MQDLH 结构, 但未转换应用程序消息数据。

应用程序可以检查 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段中返回的值以及 MQDLH 结构中返回的值, 以确定上述哪些值适用。

14. 如果格式名称为 MQFMT_XMIT_Q_HEADER，那么消息数据以 MQXQH 结构开头，可能后跟零个或多个字节的附加数据。此附加数据通常是应用程序消息数据（长度可能为零），但在附加数据开始时还可能存在一个或多个进一步的 MQ 头结构。

MQXQH 结构必须采用队列管理器的字符集和编码。MQXQH 结构之后的数据的格式，字符集和编码由 MQXQH 中包含的 MQMD 结构中的 *Format*、*CodedCharSetId* 和 *Encoding* 字段提供 MQXQH。对于存在的每个后续 MQ 头结构，该结构中的 *Format*、*CodedCharSetId* 和 *Encoding* 字段描述遵循该结构的数据；该数据是另一个 MQ 头结构或应用程序消息数据。

如果为 MQFMT_XMIT_Q_HEADER 消息指定了 MQGMO_CONVERT 选项，那么将转换应用程序消息数据和某些 MQ 头结构，但 MQXQH 结构中的数据不是。从 MQGET 调用返回时，因此：

- *MsgDesc* 参数中 *Format*、*CodedCharSetId* 和 *Encoding* 字段的值描述 MQXQH 结构中的数据，而不描述应用程序消息数据；因此，这些值与发出 MQGET 调用的应用程序指定的值不相同。

这样做的结果是，在每次 MQGET 调用之前，重复从指定了 MQGMO_CONVERT 选项的传输队列获取消息的应用程序必须将 *MsgDesc* 参数中的 *CodedCharSetId* 和 *Encoding* 字段重置为应用程序消息数据所需的值。

- 提供的最后一个 MQ 头结构中的 *Format*、*CodedCharSetId* 和 *Encoding* 字段的值描述了应用程序消息数据。如果不存在其他 MQ 头结构，那么应用程序消息数据由 MQXQH 结构中的 MQMD 结构中的这些字段描述。如果转换成功，那么值将与发出 MQGET 调用的应用程序在 *MsgDesc* 参数中指定的值相同。

如果消息是分发表消息，那么 MQXQH 结构后跟 MQDH 结构（加上其 MQOR 和 MQPMR 记录数组），而 MQDH 结构可能后跟零个或多个进一步的 MQ 头结构以及零个或多个字节的应用程序消息数据。与 MQXQH 结构一样，MQDH 结构必须采用队列管理器的字符集和编码，并且不会在 MQGET 调用上进行转换，即使指定了 MQGMO_CONVERT 选项也是如此。

上述 MQXQH 和 MQDH 结构的处理主要供消息通道代理程序在从传输队列获取消息时使用。

报告消息的转换

通常，根据原始消息的发送方指定的报告选项，报告消息可以包含不同数量的应用程序消息数据。但是，活动报告可以包含数据，但不包含在常量中提及 *_WITH_DATA 的报告选项。

尤其是，报告消息可以包含下列其中一项：

1. 无应用程序消息数据
2. 来自原始消息的部分应用程序消息数据

当原始消息的发送方指定 MQRO_*_WITH_DATA 并且消息长度超过 100 字节时，会发生此情况。

3. 来自原始消息的所有应用程序消息数据

当原始消息的发送方指定 MQRO_*_WITH_FULL_DATA 或指定 MQRO_*_WITH_DATA 并且消息为 100 字节或更短时，会发生此情况。

当队列管理器或消息通道代理程序生成报告消息时，它会将格式名称从原始消息复制到报告消息的控制信息中的 *Format* 字段。因此，报告消息中的格式名可能暗示数据长度与报告消息中实际存在的长度（上面的案例 1 和 2）不同。

如果在检索报告消息时指定了 MQGMO_CONVERT 选项：

- 对于上面的案例 1，不会调用数据转换出口（因为报告消息没有数据）。
- 对于上面的案例 3，格式名称正确表示消息数据的长度。
- 但是对于上面的情况 2，将调用数据转换出口以转换比格式名称所隐含的长度短的消息。

此外，传递到出口的原因码通常为 MQRC_NONE（即，原因码未指示消息已截断）。发生此情况的原因是报告消息的发送方截断了消息数据，而不是接收方的队列管理器截断了消息数据以响应 MQGET 调用。

由于存在这些可能性，数据转换出口不得使用格式名称来推断传递给它的数据的长度；相反，出口必须检查提供的数据的长度，并准备转换小于格式名称所隐含的长度的数据。如果可以成功转换数据，那么出口必须返回完成代码 MQCC_OK 和原因码 MQRC_NONE。要转换的消息数据的长度将作为 *InBufferLength* 参数传递到出口。

MQDXP-数据转换出口参数

MQDXP 结构是队列管理器在调用出口以在 MQGET 调用处理过程中转换消息数据时传递到数据转换出口的参数。有关数据转换出口的详细信息，请参阅 MQ_DATA_CONV_EXIT 调用的描述。

MQDXP 中的字符数据位于本地队列管理器的字符集中；这是由 *CodedCharSetId* queue-manager 属性提供的。MQDXP 中的数字数据采用本机编码；由 MQENC_NATIVE 提供。

出口只能更改 MQDXP 中的 *DataLength*、*CompCode*、*Reason* 和 *ExitResponse* 字段；将忽略对其他字段的更改。但是，如果要转换的消息是仅包含部分逻辑消息的段，那么无法更改 *DataLength* 字段。

当控制从出口返回到队列管理器时，队列管理器会检查 MQDXP 中返回的值。如果返回的值无效，那么队列管理器将继续处理，就像出口在 *ExitResponse* 中返回了 MQXDR_CONVERSION_FAILED 一样；但是，队列管理器将忽略出口在此情况下返回的 *CompCode* 和 *Reason* 字段的值，并改为使用这些字段在出口的输入上具有的值。MQDXP 中的以下值导致发生此处理：

- *ExitResponse* 字段不是 MQXDR_OK，不是 MQXDR_CONVERSION_FAILED
- *CompCode* 字段不是 MQCC_OK，也不是 MQCC_WARNING
- 当要转换的消息是仅包含部分逻辑消息的段时，*DataLength* 字段小于零或 *DataLength* 字段已更改。

下表汇总了结构中的字段。

| 表 579: MQDXP 中的字段 | | |
|-----------------------|-----------------------|--------------------------------|
| 字段 | 描述 | Topic |
| <i>StrucId</i> | 结构标识 | StrucId |
| <i>Version</i> | 结构版本号 | 版本 |
| <i>AppOptions</i> | application options | AppOptions |
| <i>Encoding</i> | 应用程序所需的数字编码 | 编码 |
| <i>CodedCharSetId</i> | 应用程序所需的字符集 | CodedCharSetId |
| <i>DataLength</i> | 消息数据的长度 (以字节计) | DataLength |
| <i>CompCode</i> | 完成代码 | CompCode |
| <i>Reason</i> | 原因码限定 <i>CompCode</i> | 原因 |
| <i>ExitResponse</i> | 来自出口的响应 | ExitResponse |
| <i>Hconn</i> | 连接句柄 | Hconn |
| <i>pEntryPoints</i> | MQIEP 结构的地址 | pEntry 点 |

字段

MQDXP 结构包含以下字段；这些字段按字母顺序进行描述。

AppOptions

类型 :MQLONG

这是由发出 MQGET 调用的应用程序指定的 MQGMO 结构的 *Options* 字段的副本。出口可能需要检查这些信息以确定是否指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项。

这是出口的输入字段。

CodedCharSetId

类型 :MQLONG

这是发出 MQGET 调用的应用程序所需的字符集的编码字符集标识; 请参阅 MQMD 结构中的 *CodedCharSetId* 字段以获取更多详细信息。如果应用程序在 MQGET 调用上指定了特殊值 MQCCSI_Q_MGR, 那么在调用出口之前, 队列管理器会将此值更改为队列管理器使用的字符集的实际字符集标识。

如果转换成功, 那么出口必须将其复制到消息描述符中的 *CodedCharSetId* 字段。

这是出口的输入字段。

CompCode

类型: MQLONG

调用出口时, 这包含返回到发出 MQGET 调用的应用程序的完成代码 (如果该出口未执行任何操作)。它始终是 MQCC_WARNING, 因为该消息已被截断, 或者该消息需要转换, 但尚未执行此操作。

在出口输出时, 此字段包含要在 MQGET 调用的 *CompCode* 参数中返回到应用程序的完成代码; 只有 MQCC_OK 和 MQCC_WARNING 有效。请参阅 *Reason* 字段的描述, 以获取有关出口如何在输出上设置此字段的建议。

这是出口的输入/输出字段。

DataLength

类型: MQLONG

调用出口时, 此字段包含应用程序消息数据的原始长度。如果截断消息以适合应用程序提供的缓冲区, 那么提供给出口的消息大小小于 *DataLength* 的值。提供给出口的消息大小始终由出口的 *InBufferLength* 参数提供, 而不考虑已发生的任何截断。

截断由 *Reason* 字段指示, 在输入到出口时, 该字段的值为 MQRC_TRUNCATED_MSG_RECEIVED。

大多数转换不需要更改此长度, 但如果需要, 出口可以这样做; 出口设置的值将返回到 MQGET 调用的 *DataLength* 参数中的应用程序。但是, 如果要转换的消息是仅包含部分逻辑消息的段, 那么不能更改此长度。这是因为更改长度将导致逻辑消息中的后续段的偏移量不正确。

请注意, 如果出口想要更改数据的长度, 请注意队列管理器已根据未转换数据的长度决定消息数据是否适合应用程序的缓冲区。此决策确定是从队列中除去消息 (还是对浏览请求移动了浏览光标), 并且不受转换所导致的数据长度的任何更改影响。因此, 建议转换出口不会导致应用程序消息数据的长度发生更改。

如果字符转换确实意味着长度的更改, 那么可以将字符串转换为另一个长度相同的字符串 (以字节为单位), 截断尾部空格或根据需要填充空格。

如果消息不包含应用程序消息数据, 那么不会调用出口; 因此 *DataLength* 始终大于零。

这是出口的输入/输出字段。

Encoding

类型: MQLONG

应用程序所需的数字编码。

这是发出 MQGET 调用的应用程序所需的数字编码; 请参阅 MQMD 结构中的 *Encoding* 字段以获取更多详细信息。

如果转换成功, 那么出口会将其复制到消息描述符中的 *Encoding* 字段。

这是出口的输入字段。

ExitOptions

类型: MQLONG

这是保留字段; 其值为 0。

ExitResponse

类型: MQLONG

来自出口的响应。此值由出口设置以指示转换是否成功。它必须是下列其中一项:

MQXDR_OK

转换成功。

如果出口指定此值，那么队列管理器将向发出 MQGET 调用的应用程序返回以下内容：

- 出口输出中 *CompCode* 字段的值
- 出口输出中 *Reason* 字段的值
- 出口输出中 *DataLength* 字段的值
- 出口的输出缓冲区 *OutBuffer* 的内容。返回的字节数是出口的 *OutBufferLength* 参数的较小值，也是出口输出的 *DataLength* 字段的值。

如果出口的消息描述符参数中的 *Encoding* 和 *CodedCharSetId* 字段都未更改，那么队列管理器将返回：

- 出口的输入上 MQDXP 结构中 *Encoding* 和 *CodedCharSetId* 字段的值。

如果出口的消息描述符参数中的一个或两个 *Encoding* 和 *CodedCharSetId* 字段已更改，那么队列管理器将返回：

- 出口输出时出口的消息描述符参数中 *Encoding* 和 *CodedCharSetId* 字段的值

MQXDR_CONVERSION_FAILED

转换失败。

如果出口指定此值，那么队列管理器将向发出 MQGET 调用的应用程序返回以下内容：

- 出口输出中 *CompCode* 字段的值
- 出口输出中 *Reason* 字段的值
- 出口的输入上 *DataLength* 字段的值
- 出口的输入缓冲区 *InBuffer* 的内容。返回的字节数由 *InBufferLength* 参数给出

如果出口已变更 *InBuffer*，那么结果未定义。

ExitResponse 是出口的输出字段。

Hconn

类型 :MQHCONN

这是可用于 MQXCNVC 调用的连接句柄。此句柄不一定与发出 MQGET 调用的应用程序指定的句柄相同。

pEntryPoints

类型 :PMQIEP

MQIEP 结构的地址，可通过该结构进行 MQI 和 DCI 调用。

Reason

类型 :MQLONG

原因码限定 *CompCode*。

调用出口时，如果出口选择不执行任何操作，那么这将包含返回到发出 MQGET 调用的应用程序的原因码。可能的值包括 MQRC_TRUNCATED_MSG_RECEIVED (指示消息已截断以适合应用程序提供的缓冲区) 和 MQRC_NOT_CONVERSION (指示消息需要转换但尚未执行此操作)。

在出口输出时，此字段包含要在 MQGET 调用的 *Reason* 参数中返回到应用程序的原因；建议执行以下操作：

- 如果 *Reason* 在输入到出口时具有值 MQRC_TRUNCATED_MSG_RECEIVED，那么无论转换成功还是失败，都不得变更 *Reason* 和 *CompCode* 字段。

(如果 *CompCode* 字段不是 MQCC_OK，那么检索消息的应用程序可以通过将消息描述符中返回的 *Encoding* 和 *CodedCharSetId* 值与请求的值进行比较来识别转换失败；相反，应用程序无法将截断的消息与适合缓冲区的消息区分开来。因此，必须首先返回 MQRC_TRUNCATED_MSG_RECEIVED，而不返回指示转换失败的任何原因。)

- 如果 *Reason* 在输入到出口时具有任何其他值：

– 如果转换成功，那么必须将 *CompCode* 设置为 MQCC_OK，并将 *Reason* 设置为 MQRC_NONE。

- 如果转换失败，或者消息扩展并且必须截断以适合缓冲区，那么 *CompCode* 必须设置为 MQCC_WARNING (或保持不变)，并且 *Reason* 设置为列出的其中一个值，以指示失败的性质。
请注意，如果转换后的消息对于缓冲区过大，那么仅当发出 MQGET 调用的应用程序指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项时，才必须截断该消息：
 - 如果它指定了该选项，那么将返回原因 MQRC_TRUNCATED_MSG_RECEIVED。
 - 如果未指定该选项，那么将返回未转换的消息，原因码为 MQRC_CONVERTED_MSG_TOO_BIG。

建议出口使用列出的原因码来指示转换失败的原因，但是如果认为适当，出口可以从 MQRC_* 代码集中返回其他值。此外，还会分配 MQRC_APPL_FIRST 到 MQRC_APPL_LAST 的值范围，以供出口使用，以指示出口希望与发出 MQGET 调用的应用程序进行通信的条件。

注：如果无法成功转换消息，那么出口必须在 *ExitResponse* 字段中返回 MQXDR_CONVERSION_FAILED，以便队列管理器返回未转换的消息。无论在 *Reason* 字段中返回的原因码如何，都是如此。

MQRC_APPL_FIRST

(900, X'384') 应用程序定义的原因码的最低值。

MQRC_APPL_LAST

(999, X'3E7') 应用程序定义的原因码的最高值。

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') 转换的数据对于缓冲区太大。

MQRC_NOT_汇率

(2119, X'847') 未转换消息数据。

MQRC_SOURCE_CCSID_ERROR

(2111, X'83F') 源编码字符集标识无效。

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841') 无法识别消息中的压缩十进制编码。

MQRC_SOURCE_FLOAT_ENC_ERROR

(2114, X'842') 无法识别消息中的浮点编码。

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') 无法识别源整数编码。

MQRC_TARGET_CCSID_ERROR

(2115, X'843') 目标编码字符集标识无效。

MQRC_TARGET_DECIMAL_ENC_ERROR

(2117, X'845') 接收器指定的压缩十进制编码无法识别。

MQRC_TARGET_FLOAT_ENC_ERROR

(2118, X'846') 接收器指定的浮点编码无法识别。

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') 无法识别目标整数编码。

MQRC_TRUNCATED_MSG_RECEIVED

(2079, X'81F') 已返回截断的消息 (处理已完成)。

这是出口的输入/输出字段。

StrucId

类型: MQCHAR4

结构标识。该值必须为:

MQDXP_STRUC_ID

数据转换出口参数结构的标识。

对于 C 编程语言，还定义了常量 MQDXP_STRUC_ID_ARRAY; 此值与 MQDXP_STRUC_ID 相同，但是字符数组而不是字符串。

这是出口的输入字段。

Version

类型: MQLONG

结构版本号。该值必须为:

MQDXP_VERSION_1

数据转换出口参数结构的版本号。

以下常量指定当前版本的版本号:

MQDXP_CURRENT_VERSION

当前版本的数据转换出口参数结构。

注: 当引入此结构的新版本时, 不会更改现有部件的布局。因此, 出口必须检查 *Version* 字段是否等于或大于包含出口需要使用的字段的最低版本。

这是出口的输入字段。

C 声明

```
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 声明 (仅限 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.
```

System/390 汇编程序声明

```
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) |

MQXCNCV-转换字符

MQXCNCV 调用使用 C 编程语言将字符从一个字符集转换为另一个字符集。

此调用是 WebSphere MQ 数据转换接口 (DCI) 的一部分，DCI 是 WebSphere MQ 框架接口之一。

注: 可以从应用程序和数据转换出口环境使用调用。

语法

MQXCNCV (*Hconn*, 选项, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

参数

Hconn

类型:MQHCONN-输入

此句柄表示与队列管理器的连接。

在数据转换出口中，*Hconn* 通常是传递到 MQDXP 结构的 *Hconn* 字段中的数据转换出口的句柄; 此句柄不一定与发出 MQGET 调用的应用程序指定的句柄相同。

在 IBM i 上，可以为 *Hconn* 指定以下特殊值:

MQHC_DEF_HCONN

缺省连接句柄。

如果运行 CICS TS 3.2 或更高版本的应用程序，请确保将调用 MQXCNCV 调用的字符转换出口程序定义为 OPENAPI。此定义防止因连接不正确导致 2018 MQRC_HCONN_ERROR 错误，并且允许完成 MQGET。

options

类型:MQLONG-输入

用于控制 MQXCNCV 的操作的选项。

可以指定所描述的零个或多个选项。如果需要多个值，那么这些值可以是:

- 相加 (不要将同一常量相加多次)，或者
- 使用按位 OR 运算进行组合 (如果编程语言支持位运算)

缺省-转换选项: 以下选项控制缺省字符转换的使用:

MQDCC_DEFAULT_CONVERSION

缺省转换。

此选项指定如果调用上指定的一个或两个字符集不受支持，那么可以使用缺省字符转换。这允许队列管理器在转换字符串时使用安装指定的缺省字符集，该字符集近似指定的字符集。

注: 使用近似字符集来转换字符串的结果是可以不正确地转换某些字符。这可以通过在字符串中仅使用对指定字符集和缺省字符集都通用的字符来避免。

缺省字符集由配置选项在安装或重新启动队列管理器时定义。

如果未指定 MQDCC_DEFAULT_CONVERT，那么队列管理器仅使用指定的字符集来转换字符串，如果其中一个或两个字符集不受支持，那么调用将失败。

此选项在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux 和 Windows。

填充选项: 以下选项允许队列管理器用空格填充转换后的字符串或废弃无关紧要的尾部字符，以使转换后的字符串适合目标缓冲区:

MQDCC_FILL_TARGET_BUFFER

填充目标缓冲区。

此选项请求以完全填充目标缓冲区的方式进行转换:

- 如果字符串在转换时收缩，那么将添加尾部空格以填充目标缓冲区。
- 如果字符串在转换时扩展，那么将废弃不重要的尾部字符，以使转换后的字符串适合目标缓冲区。如果可以成功完成此操作，那么将使用 MQCC_OK 和原因码 MQRC_NONE 完成调用。

如果无意义的尾部字符太少，那么会将尽可能多的字符串放在目标缓冲区中，并且会使用 MQCC_WARNING 和原因码 MQRC_CONVERTED_MSG_TOO_BIG 完成调用。

不重要的字符包括:

- 尾部空格
- 字符串中第一个空字符后的字符 (但不包括第一个空字符本身)
- 如果字符串 *TargetCCSID* 和 *TargetLength* 导致无法使用有效字符完全设置目标缓冲区，那么调用将失败并返回 MQCC_FAILED 和原因码 MQRC_TARGET_LENGTH_ERROR。当 *TargetCCSID* 是纯 DBCS 字符集 (例如 UCS-2)，但 *TargetLength* 指定的长度为奇数字节时，可能会发生此情况。
- *TargetLength* 可以小于或大于 *SourceLength*。从 MQXCNVC 返回时，*DataLength* 具有与 *TargetLength* 相同的值。

如果未指定此选项:

- 允许字符串根据需要在目标缓冲区中收缩或展开。不会添加或废弃无意义的尾部字符。

如果转换后的字符串适合目标缓冲区，那么调用将使用 MQCC_OK 完成，原因码为 MQRC_NONE。

如果转换后的字符串对于目标缓冲区过大，那么会在目标缓冲区中放置任意数量的字符串，并且调用会完成 MQCC_WARNING 和原因码 MQRC_CONVERTED_MSG_TOO_BIG。请注意，在这种情况下，可以返回少于 *TargetLength* 个字节的字节。

- *TargetLength* 可以小于或大于 *SourceLength*。从 MQXCNVC 返回时，*DataLength* 小于或等于 *TargetLength*。

此选项在以下环境中受支持: AIX, HP-UX, IBM i, Solaris, Linux 和 Windows。

编码选项: 描述的选项可用于指定源字符串和目标字符串的整数编码。当相应的字符集标识指示主存储器中字符集表示依赖于用于二进制整数的编码时，仅使用相关编码。这仅影响某些多字节字符集 (例如，UCS-2 字符集)。

如果字符集是单字节字符集 (SBCS) 或在主存储器中具有不依赖于整数编码的表示的多字节字符集，那么将忽略该编码。

必须仅指定其中一个 MQDCC_SOURCE_* 值，并与其中一个 MQDCC_TARGET_* 值组合:

MQDCC_SOURCE_ENC_NATIVE

源编码是环境和编程语言的缺省值。

MQDCC_SOURCE_ENC_NORMAL

源编码正常。

MQDCC_SOURCE_ENC_逆向

源编码已反转。

MQDCC_SOURCE_ENC_UNDEFINED

未定义源编码。

MQDCC_TARGET_ENC_NATIVE

目标编码是环境和编程语言的缺省值。

MQDCC_TARGET_ENC_NORMAL

目标编码正常。

MQDCC_TARGET_ENC_逆向

目标编码已反转。

MQDCC_TARGET_ENC_UNDEFINED

目标编码未定义。

可以将先前定义的编码值直接添加到 *Options* 字段。但是，如果从 MQMD 或其他结构中的 *Encoding* 字段获取源或目标编码，那么必须执行以下处理：

1. 必须通过消除 float 和 packed-decimal 编码从 *Encoding* 字段中抽取整数编码；请参阅 [第 787 页的『分析编码』](#) 以获取有关如何执行此操作的详细信息。
2. 在添加到 *Options* 字段之前，必须将步骤 1 生成的整数编码乘以相应的因子。这些因素包括：
 - 源编码的 MQDCC_SOURCE_ENC_FACTOR
 - 用于目标编码的 MQDCC_TARGET_ENC_FACTOR

以下示例代码说明了如何使用 C 编程语言对其进行编码：

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_SOURCE_ENC_FACTOR
          + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_TARGET_ENC_FACTOR;
```

如果未指定，那么编码选项缺省为 undefined (MQDCC_*_ENC_UNDEFINED)。在大多数情况下，这不会影响 MQXCNCV 调用的成功完成。但是，如果对应的字符集是具有依赖于编码的表示的多字节字符集（例如，UCS-2 字符集），那么调用将失败，原因码为 MQRC_SOURCE_INTEGER_ENC_ERROR 或 MQRC_GET_TARINTEGER_ENC_ERROR（视情况而定）。

编码选项在以下环境中受支持：AIX，HP-UX，z/OS，IBM i，Solaris，Linux 和 Windows。

缺省选项：如果未指定任何先前描述的选项，那么可以使用以下选项：

MQDCC_NONE

未指定任何选项。

MQDCC_NONE 定义为帮助程序文档。不打算将此选项与任何其他选项一起使用，但由于其值为零，因此无法检测到此类使用。

SourceCCSID

类型：MQLONG-输入

这是 *SourceBuffer* 中输入字符串的编码字符集标识。

SourceLength

类型：MQLONG-输入

这是 *SourceBuffer* 中输入字符串的长度（以字节为单位）；必须大于或等于 0。

SourceBuffer

类型：MQCHARxSource 长度-输入

这是包含要从一个字符集转换为另一个字符集的字符串的缓冲区。

TargetCCSID

类型：MQLONG-输入

这是要将 *SourceBuffer* 转换为的字符集的编码字符集标识。

TargetLength

类型：MQLONG-输入

这是输出缓冲区 *TargetBuffer* 的长度 (以字节计); 它必须为零或更大。它可以小于或大于 *SourceLength*。

TargetBuffer

类型: MQCHARxTarget 长度-输出

这是将字符串转换为 *TargetCCSID* 定义的字符集后的字符串。转换后的字符串可以比未转换的字符串短或长。 *DataLength* 参数指示返回的有效字节数。

DataLength

类型: MQLONG - 输出

这是输出缓冲区 *TargetBuffer* 中返回的字符串的长度。转换后的字符串可以比未转换的字符串短或长。

CompCode

类型: MQLONG - 输出

它是下列项之一:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') 转换的数据对于缓冲区太大。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') 数据长度参数无效。

MQRC_DBCS_ERROR

(2150, X'866') DBCS 字符串无效。

MQRC_HCONN_ERROR

(2018, X'7E2') 连接句柄无效。

MQRC_OPTIONS_ERROR

(2046, X'7FE') 选项无效或不一致。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 没有足够系统资源可用。

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861') 源缓冲区参数无效。

MQRC_SOURCE_CCSID_ERROR

(2111, X'83F') 源编码字符集标识无效。

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') 无法识别源整数编码。

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') 源长度参数无效。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 没有足够的存储空间可用。

MQRC_TARGET_BUFFER_ERROR

(2146, X'862 ') 目标缓冲区参数无效。

MQRC_TARGET_CCSID_ERROR

(2115, X'843 ') 目标编码字符集标识无效。

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844 ') 无法识别目标整数编码。

MQRC_TARGET_LENGTH_ERROR

(2144, X'860 ') 目标长度参数无效。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 发生了意外错误。

有关这些代码的详细信息，请参阅[原因码](#)。

C 调用

```
MQXCNCV (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
         TargetCCSID, TargetLength, TargetBuffer, &DataLength,
         &CompCode, &Reason);
```

按如下所示声明参数：

```
MQHCONN  Hconn;           /* Connection handle */
MQLONG   Options;        /* Options that control the action of
MQXCNCV */
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 声明 (仅限 IBM i)

```
CALL 'MQXCNCV' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
                    SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
                    TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

按如下所示声明参数：

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Options that control the action of MQXCNCV
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 汇编程序声明

```

CALL MQXCNCV, (HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,          X
              SOURCEBUFFER, TARGETCCSID, TARGETLENGTH, TARGETBUFFER, X
              DATALENGTH, COMPCODE, REASON)

```

按如下所示声明参数:

| | | | |
|----------------|----|-------|--|
| HCONN | DS | F | Connection handle |
| OPTIONS | DS | F | Options that control the action of MQXCNCV |
| 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-数据转换出口

MQ_DATA_CONV_EXIT 调用描述传递到数据转换出口的参数。

队列管理器未提供名为 MQ_DATA_CONV_EXIT 的入口点 (请参阅用法说明 11)。

此定义是 WebSphere MQ 数据转换接口 (DCI) 的一部分, DCI 是 WebSphere MQ 框架接口之一。

语法

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

参数

DataConvExitParms

类型:MQDXP-输入/输出

此结构包含与出口调用相关的信息。出口设置此结构中的信息以指示转换结果。请参阅 [第 797 页的『MQDXP-数据转换出口参数』](#) 以获取此结构中的字段的详细信息。

MsgDesc

类型:MQMD-输入/输出

在输入到出口时, 这是与 *InBuffer* 参数中传递到出口的消息数据相关联的消息描述符。

注: 传递到出口的 *MsgDesc* 参数始终是调用出口的队列管理器所支持的 MQMD 的最新版本。如果出口旨在不同环境之间可移植, 那么该出口将检查 *MsgDesc* 中的 *Version* 字段, 以验证该出口需要访问的字段是否存在于结构中。

在以下环境中, 将向出口传递 version-2 MQMD: AIX, HP-UX, IBM i, Solaris, Linux 和 Windows。在支持数据转换出口的所有其他环境中, 将向出口传递 version-1 MQMD。

在输出时, 如果转换成功, 那么出口会将 *Encoding* 和 *CodedCharSetId* 字段更改为应用程序所请求的值; 这些更改将反映回应用程序。将忽略出口对结构所作的任何其他更改; 这些更改不会反映回应用程序。

如果出口在 MQDXP 结构的 *ExitResponse* 字段中返回 MQXDR_OK，但未更改消息描述符中的 *Encoding* 或 *CodedCharSetId* 字段，那么队列管理器将为这些字段返回 MQDXP 结构中对应字段在出口输入时具有的值。

InBuffer 长度

类型: MQLONG-输入

InBuffer 的长度 (以字节计)。

这是输入缓冲区 *InBuffer* 的长度，并指定出口要处理的字节数。*InBufferLength* 是转换前消息数据的长度以及应用程序在 MQGET 调用上提供的缓冲区的长度中的较小者。

该值始终大于零。

InBuffer

类型: MQBYTExInBufferLength -输入

包含未转换消息的缓冲区。

这包含转换前的消息数据。如果出口无法转换数据，那么在出口完成后，队列管理器会将此缓冲区的内容返回给应用程序。

注: 出口不应更改 *InBuffer*; 如果此参数已更改，那么结果未定义。

在 C 编程语言中，此参数定义为一个指向 void 的指针。

OutBuffer 长度

类型: MQLONG-输入

OutBuffer 的长度 (以字节计)。

这是输出缓冲区 *OutBuffer* 的长度，并且与应用程序在 MQGET 调用上提供的缓冲区的长度相同。

该值始终大于零。

OutBuffer

类型: MQBYTExOutBufferLength -输出

包含已转换消息的缓冲区。

在出口输出时，如果转换成功 (如 *DataConvExitParms* 参数的 *ExitResponse* 字段中的值 MQXDR_OK 所指示)，那么 *OutBuffer* 将以所请求的表示法包含要传递到应用程序的消息数据。如果转换失败，那么将忽略出口对此缓冲区进行的任何更改。

在 C 编程语言中，此参数定义为一个指向 void 的指针。

使用说明

1. 数据转换出口是用户编写的出口，在处理 MQGET 调用期间接收控制。数据转换出口所执行的函数由该出口的提供程序定义；但是，该出口必须符合此处描述的规则以及相关参数结构 MQDXP 中的规则。
可用于数据转换出口的编程语言由环境确定。
2. 仅当以下所有均为 true 时，才会调用该出口：
 - 在 MQGET 调用上指定了 MQGMO_CONVERT 选项
 - 消息描述符中的 *Format* 字段不是 MQFMT_NONE
 - 消息尚未包含在必需的表示中；即，消息的 *CodedCharSetId* 和/或 *Encoding* 中的一个或两个与 MQGET 调用上提供的消息描述符中的应用程序指定的值不同
 - 队列管理器尚未成功完成转换
 - 应用程序缓冲区的长度大于零
 - 消息数据的长度大于零
 - MQGET 操作期间至今的原因码为 MQRC_NONE 或 MQRC_TRUNCATED_MSG_RECEIVED
3. 在编写出口时，请考虑以允许其转换已截断的消息的方式对该出口进行编码。可通过以下方式生成截断的消息：

- 接收应用程序提供的缓冲区小于消息，但在 MQGET 调用上指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项。

在此情况下，输入到出口的 *DataConvExitParms* 参数中的 *Reason* 字段具有值 MQRC_TRUNCATED_MSG_RECEIVED。

- 消息的发送方在发送之前将其截断。例如，报告消息可能会发生此情况 (请参阅第 796 页的『报告消息的转换』以获取更多详细信息)。

在这种情况下，出口输入的 *DataConvExitParms* 参数中的 *Reason* 字段具有值 MQRC_NONE (如果接收应用程序提供了足够大的缓冲区以用于消息)。

因此，输入到出口的 *Reason* 字段的值不能始终用于确定消息是否已截断。

截断消息的区分特征是 *InBufferLength* 参数中提供给出口的长度小于消息描述符中的 *Format* 字段中包含的格式名所隐含的长度。因此，在尝试转换任何数据之前，出口应检查 *InBufferLength* 的值；出口不应假定已提供格式名称所隐含的全部数据量。

如果未写入出口以转换截断的消息，并且 *InBufferLength* 小于预期值，那么该出口将在 *DataConvExitParms* 参数的 *ExitResponse* 字段中返回 MQXDR_CONVERSION_FAILED，并将 *CompCode* 和 *Reason* 字段设置为 MQCC_WARNING 和 MQRC_FORMAT_ERROR。

如果已写入出口以转换截断的消息，那么出口将转换尽可能多的数据 (请参阅下一个使用说明)，请注意不要尝试在 *InBuffer* 结束后检查或转换数据。如果转换成功完成，那么出口将使 *DataConvExitParms* 参数中的 *Reason* 字段保持不变。如果消息被接收方的队列管理器截断，那么这将返回 MQRC_TRUNCATED_MSG_RECEIVED；如果消息被消息发送方截断，那么返回 MQRC_NONE。

消息还可以展开期间转换，使其大于 *OutBuffer*。在这种情况下，出口必须决定是否截断消息；*DataConvExitParms* 参数中的 *AppOptions* 字段指示接收应用程序是否指定了 MQGMO_ACCEPT_TRUNCATED_MSG 选项。

- 通常，将转换提供给 *InBuffer* 中出口的消息中的所有数据，或者不转换任何数据。但是，如果消息在转换之前或转换期间被截断，那么会发生此异常；在这种情况下，缓冲区末尾可能有一个不完整的项 (例如：双字节字符的 1 字节，或 4 字节整数的 3 字节)。在这种情况下，请考虑省略不完整的项，并将 *OutBuffer* 中未使用的字节设置为空。但是，应该转换数组或字符串中的完整元素或字符。
- 首次需要出口时，队列管理器会尝试装入与格式同名的对象 (扩展名除外)。装入的对象必须包含用于处理具有该格式名称的消息的出口。请考虑使出口名称与包含出口的对象名称相同，尽管并非所有环境都需要此名称。
- 由于应用程序已连接到了队列管理器，因此当应用程序尝试检索使用该 *Format* 的第一条消息时，会装入出口的新副本。对于 CICS 或 IMS 应用程序，这意味着当 CICS 或 IMS 子系统连接到队列管理器时。如果队列管理器已废弃先前装入的副本，那么还可以在其他时间装入新副本。因此，出口不得尝试使用静态存储器将信息从出口的一次调用传递到下一次调用-可以在两次调用之间卸载出口。
- 如果存在与队列管理器支持的其中一种内置格式同名的用户提供的出口，那么用户提供的出口不会替换内置转换例程。调用此类出口的唯一情况是：
 - 如果内置转换例程无法处理与所涉及的 *CodedCharSetId* 或 *Encoding* 之间的转换，或者
 - 如果内置转换例程未能转换数据 (例如，因为存在无法转换的字段或字符)。
- 出口的作用域与环境相关。必须选择 *Format* 名称以将与其他格式冲突的风险降至最低。请考虑从用于标识定义格式名称的应用程序的字符开始。
- 数据转换出口在发出 MQGET 调用的程序的环境中运行；环境包括地址空间和用户概要文件 (如果适用)。程序可以是将消息发送到不支持消息转换的目标队列管理器的消息通道代理。该出口无法损害队列管理器的完整性，因为它不会在队列管理器的环境中运行。
- 出口可以使用的唯一 MQI 调用是 MQXCNV; 尝试使用其他 MQI 调用失败，原因码为 MQRC_CALL_IN_PROGRESS 或其他不可预测的错误。
- 队列管理器未提供名为 MQ_DATA_CONV_EXIT 的入口点。但是，为 C 编程语言中的名称 MQ_DATA_CONV_EXIT 提供了 typedef，这可用于声明用户编写的出口，以确保参数正确。出口的名称必须与格式名称 (MQMD 中的 *Format* 字段中包含的名称) 相同，尽管这在所有环境中都不是必需的。

以下示例说明如何使用 C 编程语言声明用于处理格式 MYFORMAT 的出口:

```
#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. 在 z/OS 上, 如果 API 交叉出口也生效, 那么将在数据转换出口之后调用该出口。

C 调用

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
          InBuffer, OutBufferLength, OutBuffer);
```

传递到出口的参数声明如下:

```
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 声明 (仅限 IBM i)

```
CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                      INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

传递到出口的参数声明如下:

```
** Data-conversion exit parameter block
01 DATACONVEXITPARMS.
   COPY CMQDXPV.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Length in bytes of INBUFFER
01 INBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the unconverted message
01 INBUFFER PIC X(n).
** Length in bytes of OUTBUFFER
01 OUTBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the converted message
01 OUTBUFFER PIC X(n).
```

System/390 汇编程序声明

```
CALL EXITNAME, (DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH, X
               INBUFFER, OUTBUFFERLENGTH, OUTBUFFER)
```

传递到出口的参数声明如下:

```

DATAONVEXITPARMS  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
    
```

指定为 MQRFH2 元素的属性

可以将非消息描述符属性指定为 MQRFH2 头文件夹中的元素。指定为属性的 MQRFH2 元素的概述。

这将保留与先前版本的 WebSphere MQ JMS 和 XMS 客户机的兼容性。本节描述如何在 MQRFH2 头中指定属性。

要使用 MQRFH2 元素作为属性，请指定元素，如将 [WebSphere MQ 类用于 Java](#) 中所述。本信息补充了第 457 页的『MQRFH2 -规则和格式化头 2』中描述的信息。

将属性数据类型映射到 MQRFH2 数据类型

本主题提供有关映射到相应 MQRFH2 数据类型的消息属性类型的信息。

| 消息属性类型 | MQRFH2 数据类型 |
|-----------|-------------|
| MQBYTE [] | bin.hex |
| MQBOOL | 布尔值 |
| MQINT8 | i1 |
| MQINT16 | i2 |
| MQINT32 | i4 |
| MQINT64 | i8 |
| MQFLOAT32 | r4 |
| MQFLOAT64 | r8 |
| MQCHAR [] | 字符串 |

假定没有数据类型的任何元素的类型为 "string"。

MQRFH2 数据类型 int(表示未指定大小的整数) 被视为 i8。

元素属性 `xsi:nil='true'` 指示空值。请勿将属性 `xsi:nil='false'` 用于非空值。

例如，以下属性具有空值:

```
<NullProperty xsi:nil='true'></NullProperty>
```

字节或字符串属性可以具有空值。这由具有零长度元素值的 MQRFH2 元素表示。

例如，以下属性具有空值:

```
<EmptyProperty></EmptyProperty>
```

受支持的 MQRFH2 文件夹

使用消息描述符字段作为属性的概述。

MQRFH2 头和 JMS 中描述了 <jms>, <mcd>, <mqext>和 <usr> 文件夹。 <usr> 文件夹用于传输与消息关联的任何 JMS 应用程序定义的属性。 <usr> 文件夹中不允许使用组。

MQRFH2 头和 JMS 支持以下其他文件夹:

- <mq>

此文件夹用于并保留供 IBM WebSphere MQ 使用的 MQ 定义的属性。

- <mq_usr>

此文件夹可用于传输未公开为 JMS 用户定义属性的任何应用程序定义属性, 因为这些属性可能不满足 JMS 属性的需求。 此文件夹可以包含 <usr> 文件夹无法包含的组。

- 使用 content='properties' 属性标记的任何文件夹。

这样的文件夹相当于内容中的 <mq_usr> 文件夹。

- <mqps>

此文件夹用于 IBM WebSphere MQ 发布/预订属性。

IBM WebSphere MQ 还支持 WAS/SIB 已在使用的以下文件夹:

- <sib>

此文件夹用于 WAS/SIB 系统消息属性, 这些属性未公开为 JMS 属性, 或映射到 JMS_IBM_* 属性, 但公开给 WAS/SIB 应用程序; 这些属性包括正向和反向路由路径属性。

至少某些属性不能公开为 JMS 属性, 因为它们是字节数组。 如果应用程序将属性添加到此文件夹, 那么将忽略或删除该值。

- <sib_usr>

此文件夹用于并保留 WAS/SIB 用户消息属性, 这些属性不能作为 JMS 用户属性公开, 因为它们不是受支持的类型; 它们向 WAS/SIB 应用程序公开。

这些是用户属性, 您可以通过 SIMessage 接口获取或设置这些属性, 但字节数组的内容会映射到必需的属性值。

如果 IBM WebSphere MQ 应用程序将任意 bin.hex 元素写入文件夹, 那么该应用程序可能会接收到 IOException, 因为它的格式不是期望复原的格式。 如果添加除 bin.hex 元素以外的任何内容, 那么将接收到 ClassCastException。

请勿尝试使用此文件夹使属性可供 WAS/SIB 使用; 而是使用 <usr> 文件夹来实现此目的。

- <sib_context>

此文件夹用于未向 WAS/SIB 用户应用程序公开或未作为 JMS 属性公开的 WAS/SIB 系统消息属性。 这些属性包括用于 Web Service 和类似服务的安全性和事务性属性。

应用程序不得向此文件夹添加属性。

- <mqema>

此文件夹由 WAS/SIB 而不是 <mqext> 文件夹使用。

MQRFH2 文件夹名称区分大小写。

保留以下文件夹, 以小写或大写字母混合使用:

- 以 mq 或 wmq 为前缀的任何文件夹; 保留供 IBM WebSphere MQ 使用。
- 以 sib 为前缀的任何文件夹; 保留供 WAS/SIB 使用。
- <Root> 和 <Body> 文件夹; 保留但未使用。

以下文件夹无法识别为包含消息属性:

- <psc>

由 WebSphere Message Broker 用于将发布/预订命令消息传递到代理。

- <pscr>

由 WebSphere Message Broker 用于包含来自代理的信息, 以响应发布/预订命令消息。

- WebSphere Message Broker 未定义任何未使用 content= 'properties' 属性标记的文件夹。

请勿在 <psc> 或 <pscr> 文件夹上指定 content= 'properties'。如果执行此操作，那么会将这些文件夹视为属性，并且 WebSphere Message Broker 可能会按预期停止运行。

如果应用程序正在使用属性构建消息，那么在要识别为包含属性的 MQRFH2 头的 MQRFH2 头中，该头必须位于可链接到消息头的头列表中。

MQRFH2 可以前置任意数量的 MQH 标准头，或者 MQCIH，MQDLH，MQIIH，MQTM，MQTMC2 或 MQXQH。字符串或 MQCFH 结束解析，因为无法链接这些字符串或 MQCFH。

消息可能包含多个带有消息属性的 MQRFH2 头。除非另有限制 (例如 WAS/SIB)，否则具有相同名称的文件夹可以共存于不同的头中。如果这些文件夹都位于重要的头中，那么它们将被视为一个逻辑文件夹。

虽然无法将重要头中的文件夹与非重要头中的文件夹合并，但可以合并重要头中具有相同名称的文件夹，从而除去任何冲突属性。应用程序不得依赖于其消息中的属性布局。

将针对用户定义的文件夹 (即，<wmq>，<jms>，<mcd>，<usr>，<mqext>，<sib>，<sib_usr>，<sib_context>和 <mqema> 文件夹) 中的属性解析 MQRFH2 组。

将针对属性解析 IBM 定义的属性文件夹中的组 (<wmq> 和 <mq> 文件夹除外)。

MQRFH2 文件夹不能包含混合内容; 文件夹或组可以包含组或属性，也可以包含值，但不能同时包含两者。

消息的段 (第一个或后续段) 不能包含除消息描述符中的那些属性以外的 IBM WebSphere MQ 定义的属性。因此，使用 MQMF_SEGMENT 或 MQMF_SEGMENTATION_ALLOWED 设置来放置包含此类属性的消息会导致 PUT 失败并返回 MQRC_SEGMENTATION_NOT_ALLOWED。

但是，消息组可以包含 IBM WebSphere MQ 定义的属性。

生成 MQRFH2 头

如果 WebSphere MQ 将消息属性转换为其 MQRFH2 表示，那么必须将 MQRFH2 添加到消息中。它将 MQRFH2 添加为单独的头，或者将其与现有头合并。

WebSphere MQ 生成新的 MQRFH2 头可能会中断消息中的现有头。为头解析消息缓冲区的应用程序必须知道，在某些情况下，头在缓冲区中的数量和位置可能会更改。WebSphere MQ 尝试通过将消息属性合并到现有 MQRFH2 头 (可以在该头中使用) 来最大限度降低向消息添加属性的影响。它还尝试通过将生成的 MQRFH2 插入到相对于消息缓冲区中其他头的固定位置来最小化影响。

生成的 MQRFH2 头位于 MQMD 以及任意数量的 MQXQH，MQRFH 和 MQDLH 头之后，无论它们的顺序如何。生成的 MQRFH2 头紧跟在不是 MQMD，MQXQH，MQDLH 或 MQRFH 头的第一个头之前。

用于合并生成的 MQRFH2 的规则

以下规则适用于将生成的 MQRFH2 与现有 MQRFH2 合并。生成的 MQRFH2 头将与现有 MQRFH2 头合并，如果:

1. 现有 MQRFH2 位于 WebSphere MQ 将生成的 MQRFH2 或更低版本放在头链中的相同位置。
2. 生成的属性的 CCSID 与现有 MQRFH2 的 NameValueCCSID 相同。

否则，生成的头将单独放置在缓冲区中之前描述的位置。

用于合并现有 MQRFH2 中的文件夹的规则

如果将消息属性合并到现有 MQRFH2 中，那么将扫描现有 MQRFH2 以查找与消息属性匹配的文件夹，并对其合并。如果不存在匹配的文件夹，那么会将新文件夹添加到现有文件夹的末尾。如果存在匹配的文件夹，那么将搜索该文件夹。将覆盖任何匹配的属性。任何新项都将添加到文件夹的末尾。

MQRFH2 文件夹限制

MQRFH2 头中的文件夹限制概述

MQRFH2 限制适用于以下文件夹:

- `<usr>` 文件夹中的元素名称不得以前缀 JMS 开头; 此类属性名称保留供 JMS 使用, 并且对于用户定义的属性无效。

此类元素名称不会导致 MQRFH2 解析失败, 但 WebSphere MQ 消息属性 API 无法访问此元素名称。

- `<usr>` 文件夹中的元素名称不得同时使用小写或大写, NULL, TRUE, FALSE, NOT 和, 或者介于 LIKE, IN, IS 和 ESCAPE。这些名称与 SQL 关键字匹配, 并使解析选择器更加困难, 因为 `<usr>` 是在选择器中未指定特定属性的文件夹时使用的缺省文件夹。

此类元素名称不会导致 MQRFH2 解析失败, 但 WebSphere MQ 消息属性 API 无法访问此元素名称。

- 任何被视为包含消息属性的文件夹中的元素名称不得包含句点 (.) (Unicode 字符 U+002E), 因为这在属性名称中用于指示层次结构。

此类元素名称不会导致 MQRFH2 解析失败, 但 WebSphere MQ 消息属性 API 无法访问此元素名称。

通常, 包含有效 XML 样式数据的 MQRFH2 头可由 WebSphere MQ 解析而不会发生故障, 尽管 MQRFH2 的某些元素无法通过 WebSphere MQ 消息属性 API 进行访问。

MQRFH2 元素名称冲突

MQRFH2 元素名称中冲突的概述。

只能将一个值附加到消息属性。如果尝试访问属性导致值冲突, 那么将选择一个值而不是另一个值。

如果文件夹不包含同名元素, 那么用于访问 MQRFH2 元素的 WebSphere MQ 语法允许对元素进行唯一标识。如果文件夹包含多个同名元素, 那么所使用的属性值是最接近消息头的值。

如果相同名称的两个或多个文件夹包含在同一消息中的不同重要 MQRFH2 头中, 那么这适用。

在两次设置非消息描述符属性之后处理 MQGET 调用时, 可能会产生冲突: 通过 MQSETMP 调用并直接在原始 MQRFH2 头中进行处理。

如果发生此情况, 那么 API 调用与消息关联的属性优先于消息数据中的属性, 即原始 MQRFH2 头中的属性。如果发生冲突, 那么会将其视为在逻辑上位于消息数据之前。

从属性名称到 MQRFH2 文件夹和元素名称的映射

MQRFH2 头中的属性名称与元素名称之间的差异概述。

使用最终生成 MQRFH2 头的任何已定义 API 时, 为了指定消息属性 (例如, MQ JMS), 属性名称不一定是 MQRFH2 文件夹中的元素名称。

因此, 从属性名称到 MQRFH2 元素的映射将以相反的方式进行, 同时考虑包含该元素的文件夹名称和元素名称。使用 [Java](#) 中已记录了 IBM WebSphere MQ classes for JMS 中的一些示例。

| 属性名 | MQRFH2 文件夹名称 | MQRFH2 元素名称 |
|------------------------------|--------------|----------------|
| JMSDestination | jms | Dst |
| JMSType | mcd | Type, Set, Fmt |
| xxx (用户定义, 其中 xxx 不以 JMS 开头) | usr | xxx |

因此, 当 JMS 应用程序访问 JMSDestination 属性时, 此属性映射到 `<jms>` 文件夹中的 Dst 元素。

将属性指定为 MQRFH2 元素时, IBM WebSphere MQ 定义其元素, 如下所示:

| 属性名 | MQRFH2 文件夹名称 | MQRFH2 组名 | MQRFH2 元素名称 |
|--|-----------------------------|----------------------------|-------------------------------|
| <code><Property></code> | <code><usr></code> | 不适用 | <code><Property></code> |
| <code><folder>.<Property></code> | <code><folder></code> | 不适用 | <code><Property></code> |
| <code><folder>.<group>.<Property></code> | <code><folder></code> | <code><group></code> | <code><Property></code> |

例如，当 IBM WebSphere MQ JMS 应用程序尝试访问 Property1 属性时，此属性映射到 <usr> 文件夹中的 Property1 元素。wmq.Property2 属性映射到 <wmq> 文件夹中的 Property2 属性。

如果属性名包含多个属性名。字符，所使用的 MQRFH2 元素名称是在 final 之后的名称。字符和 MQRFH2 组用于构成层次结构；允许嵌套的 MQRFH2 组。

IBM WebSphere MQ 应用程序使用在 使用 WebSphere MQ classes for Java 中定义的短名称来访问 <mcld>， <jms>和 <mqext> 文件夹中的 MQRFH2 中包含的 JMS 头和特定于提供程序的属性。

可从 <usr> 文件夹访问 JMS 用户定义的属性。IBM WebSphere MQ 应用程序可以将 <usr> 文件夹用于其应用程序属性 (如果该属性可作为其用户定义的属性之一向 JMS 应用程序显示)。

如果不可接受，请选择另一个文件夹；<wmq_usr> 文件夹将作为此类非 JMS 属性的标准位置提供。

应用程序可以指定任何 MQRFH2 文件夹并将其用于明确定义的用途，如果您注意到以下内容，那么不会在第 811 页的『指定为 MQRFH2 元素的属性』中进行记录：

1. 该文件夹可能已在使用中，或者将来可能由另一个提供对其中包含的属性的未定义访问权的应用程序使用；请参阅 属性名称 以获取建议的属性名称命名约定。
2. 先前版本的 IBM WebSphere MQ classes for JMS 或 XMS 客户机无法访问这些属性，只能访问用户定义的属性的 <usr> 文件夹
3. 必须使用值设置为 properties 的属性 content 来标记文件夹，例如，content='properties'。

第 685 页的『MQSETMP-设置消息属性』会根据需要自动添加此属性。不得将此属性添加到任何 IBM 定义的文件夹，例如 <jms> 和 <usr>。这样做会导致 IBM WebSphere MQ classes for JMS 客户机在版本 7.0 之前使用 MessageFormatException 拒绝消息。

因为 <usr> 文件夹是 <Property> 语法的属性的缺省位置，所以 IBM WebSphere MQ 应用程序和 JMS 应用程序使用相同名称访问相同的用户定义属性值。

保留文件夹名称

有几个保留文件夹名称。不能将此类名称用作文件夹前缀；例如，由于保留了 Root，因此 Root.Property1 无法访问有效属性。以下列表包含保留文件夹名称：

- 根语言
- 正文
- 属性
- 环境
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- InputLocalEnvironment
- InputDestinationList
- InputExceptionList
- OutputRoot
- OutputLocalEnvironment
- OutputDestinationList
- OutputExceptionList

将属性描述符字段映射到 MQRFH2 头

将属性转换为 MQRFH2 元素时，将使用以下元素属性来指定属性描述符的重要字段：这描述了如何将 MQPD 字段转换为 MQRFH2 元素属性。

支持

"支持属性描述符" 字段拆分为三个元素属性

- **sr** 元素属性指定 MQPD_REJECT_UNSUP_MASK 位掩码中的值。
- **sa** 元素属性指定 MQPD_ACCEPT_UNSUP_MASK 位掩码中的值。
- **sx** 元素属性指定 MQPD_ACCEPT_UNSUP_IF_XMIT_MASK 位掩码中的值。

These element attributes are only valid in the <mq> folder and are ignored if set on elements in the other folders containing properties.

| 支持值 | MQRFH2 元素属性 | MQRFH2 属性值 |
|--------------------------------|-------------|--------------|
| MQPD_SUPPORT_OPTIONAL | sa | 可选 这是缺省值。 |
| MQPD_SUPPORT_REQUIRED | sr | 必需 |
| MQPD_SUPPORT_REQUIRED_IF_LOCAL | Sx | 本地 |

Context

使用 **context** 元素属性来指示属性所属的消息上下文。仅使用一个值。此元素属性对包含属性的任何文件夹中的属性有效。

| 上下文值 | MQRFH2 属性值 |
|-------------------|----------------|
| MQPD_NO_CONTEXT | none 这是缺省值。 |
| MQPD_USER_CONTEXT | 用户 |

CopyOptions

使用 **copy** 元素属性来指示应将属性复制到其中的消息。可接受多个值；请使用逗号分隔多个值。例如，**copy='reply'** 和 **copy='publish,report'** 都有效。此元素属性对包含属性的任何文件夹中的属性有效。

注：在属性定义中，单引号或双引号是有效用法，例如 **copy='reply'** 或 **copy="report"**

| CopyOption 值 | MQRFH2 属性值 |
|-------------------|--|
| MQPD_COPY_FOR 何承天 | 转发 (forward) |
| MQPD_COPY_REPLY | 应答 |
| MQPD_COPY_REPORT | 报告 |
| MQPD_COPY_PUBLISH | 发布 |
| MQPD_COPY_ALL | all 请勿将此值与任何其他值一起指定。与另一个值一起使用时，此值优先于除 none 以外的任何值。 |

| CopyOption 值 | MQRFH2 属性值 |
|-------------------|--|
| MQPD_COPY_DEFAULT | 缺省 这是缺省值。它相当于指定了三个值 MQCOPY_FOR_ID , MQCOPY_REPORT 和 MQCOPY_PUBLISH。 请勿将此值与任何其他值一起指定。 |
| MQPD_COPY_NONE | none 请勿将此值与任何其他值一起指定。与另一个值一起使用时, 此值优先。 |

Restrictions to the <mq> MQRFH2 folder

When a message is put on to a queue, it is searched for an <mq> folder so that the message can be processed according to its MQ-defined properties. 要允许高效解析 MQ 定义的属性, 以下限制适用于该文件夹:

- Only properties in the first significant <mq> folder in the message are acted upon by MQ; properties in any other <mq> folder in the message are ignored.
- 如果文件夹采用 UTF-8, 那么文件夹中仅允许使用单字节 UTF-8 字符。文件夹中的多字节字符可能导致解析失败, 并且要拒绝消息。
- Do not include MQRFH2 groups in the <mq> folder. 属性值中存在 Unicode 字符 U+003C 将导致消息被拒绝。
- 请勿在文件夹中使用转义字符串。转义字符串被视为元素的实际值。
- 只有 Unicode 字符 U+0020 被视为文件夹中的空格。所有其他字符都被视为重要字符, 并可能导致文件夹解析失败以及要拒绝的消息。

If parsing of the <mq> folder fails, or if the folder does not observe these restrictions, the message is rejected with CompCode **MQCC_FAILED** and Reason **MQRC_RFH_RESTRICTED_FORMAT_ERR**.

MQRFH2 头无效

在 MQPUT, MQPUT1 或 MQGET 调用处理时, 可能会对消息中的任何 MQRFH2 头进行部分解析, 以检查包含哪些文件夹, 并确定这些文件夹是否包含属性。无效的 MQRFH2 头概述。

如果由于结构无效而无法成功完成消息的部分解析 (例如, StructLength 字段太小), 那么:

- 如果可以确定应用程序包含某些 WebSphere MQ V 7 选项, 那么 MQPUT 或 MQPUT1 调用将失败, 原因码为 MQRC_RFH_ERROR, 以便现有应用程序不会失败。
- MQGET 调用成功返回, 并且在提供的缓冲区中返回包含错误的 MQRFH2。

如果由于无法检测到特定文件夹是否包含属性 (例如, 文件夹以 <<jms 开头) 而导致部分解析失败, 那么在确定文件夹名称之前解析失败, 那么:

- 如果可以确定应用程序包含某些 WebSphere MQ V 7 选项, 那么 MQPUT 或 MQPUT1 调用将失败, 原因码为 MQRC_RFH_FORMAT_ERROR, 以便现有应用程序不会失败。
- MQGET 调用成功返回, 并且在提供的缓冲区中返回包含错误的 MQRFH2。
- 在队列管理器内部时, 不会由于格式错误的文件夹而拒绝消息, 但始终会将该文件夹视为其内部未包含任何属性。

消息可以流经包含此类语法错误的文件夹的队列管理器网络, 但从不进行解析和检测, 而消息中的一个或多个文件夹为:

- 有效
- 已成功解析
- 用于处理消息

因此, 无法保证检测。

如果其中一个应用程序使用第 685 页的『MQSETMP-设置消息属性』或 MQINQMP 来访问属性，那么这样做会导致对 MQRFH2 文件夹进行完全解析，从而检测到无法完成解析的错误，这由 API 调用的相应返回码指示。文件夹中没有可供应用程序使用的属性。

如果尝试完全解析 MQRFH2 文件夹，并且解析器找到无法识别的元素属性或无法识别的数据类型，那么解析将继续并成功完成，而不会发出警告；这不会构成解析错误。

代码页转换

本部分描述了代码集名称和 CCSID，本地语言，z/OS 转换，IBM i 转换以及 Unicode 转换支持。

每个本地语言部分都列出以下信息：

- 支持的本机 CCSID
- 不支持的代码页转换

信息中使用了以下术语：

-8

指示对于 HP-UX，CCSID 用于 HP-UX 定义的代码集 *roman8*

AIX

指示 WebSphere MQ for AIX

HP-UX

指示 WebSphere MQ for HP-UX

Linux

指示 WebSphere MQ for Linux for Intel 和 WebSphere MQ for Linux for zSeries

HP Integrity NonStop Server

指示 HP Integrity NonStop Server 的 WebSphere MQ

OS/400

指示 IBM i 的 WebSphere MQ

Solaris

指示 WebSphere MQ for Solaris

Windows

指示 WebSphere MQ for Windows

z/OS

指示 WebSphere MQ for z/OS

数据转换的缺省值是要在目标 (接收) 系统上执行的转换。

如果源产品支持转换，那么可以通过在源上将通道属性 CONVERT 设置为 YES 来设置通道并交换数据。

注：

1. WebSphere MQ MQI 客户机信息的转换在服务器中进行，因此服务器必须支持从客户机 CCSID 到服务器 CCSID 的转换。
2. 转换可能包括 CSD/PTF 添加到最新版本的 WebSphere MQ 的支持。请检查最新服务级别的内容，以了解是否需要安装 CSD/PTF 以启用此转换。

请参阅第 818 页的表 581，以获取某些 CCSID 编号与某些行业代码集名称之间的交叉引用。

代码集名称和 CCSID

WebSphere MQ for z/OS 提供的转换多于特定于语言的表中列出的转换。

| 代码集名称 | CCSID |
|------------|-------|
| ISO 8859-1 | 819 |
| ISO 8859-2 | 912 |

表 581: 代码集名称和 CCSID (继续)

| 代码集名称 | CCSID |
|------------------|----------------|
| 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 (欧元) | 923 |
| big5 | 950 |
| eucJP | 954 5050 33722 |
| eucKR | 970 |
| eucTW | 964 |
| eucCN | 1383 |
| pck | 943 |
| GBK | 1386 |
| koi8-r | 878 |

本地语言

此信息包含 WebSphere MQ 支持的语言。

WebSphere MQ 支持的语言为:

- [美国英语-请参阅主题 第 820 页的『美国英语』](#)
- [德语-请参阅主题 第 820 页的『德语』](#)
- [丹麦语和挪威语-请参阅主题 第 821 页的『丹麦和挪威语』](#)
- [芬兰语和瑞典语-请参阅主题 第 822 页的『芬兰语和瑞典语』](#)
- [意大利语-请参阅主题 第 823 页的『意大利语』](#)
- [西班牙语-请参阅主题 第 823 页的『西班牙语』](#)
- [英国英语/盖尔语-请参阅主题 第 824 页的『英国英语/盖尔语』](#)
- [法语-请参阅主题 第 824 页的『法语』](#)
- [多语言-请参阅主题 第 825 页的『多语种』](#)
- [葡萄牙语-请参阅主题 第 825 页的『葡萄牙语』](#)
- [冰岛语-请参阅主题 第 826 页的『冰岛语』](#)
- [东欧语言-请参阅主题 第 827 页的『东欧语言』](#)
- [西里尔文-请参阅主题 第 828 页的『西里尔文』](#)
- [爱沙尼亚语-请参阅主题 第 828 页的『爱沙尼亚语』](#)
- [拉脱维亚语和立陶宛语-请参阅主题 第 829 页的『拉脱维亚语和立陶宛语』](#)
- [Ukranian-请参阅主题 第 830 页的『乌克兰语』](#)
- [希腊语-请参阅主题 第 831 页的『希腊语』](#)

- 土耳其语-请参阅主题 [第 832 页的『土耳其语』](#)
- 希伯来语-请参阅主题 [第 832 页的『希伯来历』](#)
- Farsi-请参阅主题 [第 834 页的『波斯』](#)
- Urdu-请参阅主题 [第 834 页的『乌尔都语』](#)
- 泰语-请参阅主题 [第 834 页的『泰国语』](#)
- 老挝语-请参阅主题 [第 835 页的『老挝语』](#)
- 越南语-请参阅主题 [第 835 页的『越南语』](#)
- 日语拉丁语 SBCS-请参阅主题 [第 835 页的『日语拉丁语 SBCS』](#)
- 日语片假名 SBCS-请参阅主题 [第 837 页的『日语片假名 SBCS』](#)
- 日语日语汉字/拉丁语混合-请参阅主题 [第 838 页的『日语日语汉字/拉丁语混合』](#)
- 日语日语汉字/片假名混合-请参阅主题 [第 840 页的『日语日语汉字/片假名混合』](#)
- 韩国语-请参阅主题 [第 841 页的『韩语』](#)
- 简体中文-请参阅主题 [第 841 页的『简体中文』](#)
- 繁体中文-请参阅主题 [第 842 页的『繁体中文』](#)

美国英语

美国英语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上美式英语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| IBM i, z/OS | 37, 924, 1140 |
| AIX | 819, 923, 5348 |
| HP-UX | 819, 923, 1051 |
| Windows | 437, 850, 1252, 5348, 858 |
| HP Integrity NonStop Server, Solaris 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

IBM i

代码页:

37

不转换为代码页 923 和 858

924

不转换为代码页 437, 858, 1051, 1140, 1252, 1275 和 5348

1140

不转换为代码页 924,1051 和 1275

德语

德语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上德语的本机 CCSID:

| 平台 | 本机 CCSID |
|-------------|----------------|
| IBM i, z/OS | 273, 924, 1141 |
| AIX | 819, 923, 5348 |

| 平台 | 本机 CCSID |
|--|---------------------------|
| HP-UX | 819, 923, 1051 |
| Windows | 437, 850, 858, 1252, 5348 |
| HP Integrity NonStop Server, Solaris 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

IBM i

代码页:

273

不转换为代码页 858, 923, 924 和 1275

924

不转换为代码页 273, 437, 858, 1051, 1141, 1252, 1275 和 5348

1141

不转换为代码页 924,1051 和 1275

丹麦和挪威语

丹麦语和挪威语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上丹麦语和挪威语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| 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 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

IBM i

代码页:

277

不转换为代码页 858, 923, 924 和 1275

924

不转换为代码页 277, 858, 865, 1051, 1142, 1252, 1275 和 5348

1142

不转换为代码页 924,865, 1051 和 1275

AIX

代码页:

819

不转换为代码页 865

HP-UX

代码页:

1051

不转换为代码页 865

Windows

代码页:

865

不转换为代码页 1051 和 1275

芬兰语和瑞典语

芬兰语和瑞典语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上芬兰语和瑞典语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|--------------------------------|
| IBM i, z/OS | 278, 924, 1143 |
| AIX | 819, 923, 5348 |
| HP-UX | 819, 923, 1051 |
| Windows | 437, 850, 858, 865, 1252, 5348 |
| HP Integrity NonStop Server, Solaris 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

278

不转换为代码页 858, 923, 924 和 1275

924

不转换为代码页 278, 437, 858, 865, 1051, 1143, 1252, 1275 和 5348

1143

不转换为代码页 865, 924, 1051 和 1275

AIX

代码页:

819

不转换为代码页 865

850

不转换为代码页 865

HP-UX

代码页:

1051

不转换为代码页 865

Windows

代码页:

865

不转换为代码页 1051 和 1275

意大利语

意大利语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上意大利语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| 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 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

280

不转换为代码页 858, 923, 924 和 1275

924

不转换为代码页 280, 437, 858, 1051, 1144, 1252, 1275 和 5348

1144

不转换为代码页 924,1051 和 1275

西班牙语

西班牙语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上西班牙语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| 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 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

284

不转换为代码页 858, 923, 924 和 1275

924

不转换为代码页 284, 437, 858, 1051, 1145, 1252, 1275 和 5348

1145

不转换为代码页 924,1051 和 1275

英国英语/盖尔语

英国英语/盖尔语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上英国英语/盖尔语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| IBM i, z/OS | 285, 924, 1146 |
| AIX | 819, 923, 5348 |
| HP-UX | 819, 923, 1051 |
| Windows | 437, 850, 858, 1252, 5348 |
| HP Integrity NonStop Server, Solaris 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

285

不转换为代码页 858, 923, 924 和 1275

924

不转换为代码页 285, 437, 858, 1051, 1146, 1252, 1275 和 5348

1146

不转换为代码页 924,1051 和 1275

法语

法语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上法语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| 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 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

297

不转换为代码页 858, 923, 924, 1275 和 5348

924

不转换为代码页 297, 437, 858, 1051, 1147, 1252, 1275 和 5348

1147

不转换为代码页 924,1051 和 1275

多语种

多语言的 CCSID 和 CCSID 转换的详细信息。

下表显示了在受支持的平台上用于多语言转换的本机 CCSID:

| 平台 | 本机 CCSID |
|---|---------------------------|
| 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 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

500

不转换为代码页 858 和 923

924

不转换为代码页 437, 858, 1051, 1148, 1252, 1275 和 5348

1148

不转换为代码页 924,1051 和 1275

葡萄牙语

葡萄牙语 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上葡萄牙语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| 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 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

37

不转换为代码页 858,923 和 1275

500

不转换为代码页 858,923 和 1275

924

不转换为代码页 858 ,860 ,1051 ,1140 ,1252 ,1275 ,5348

1140

不转换为代码页 860,924,1051 和 1275

HP-UX

代码页:

1051

不转换为代码页 860

Windows

代码页:

860

不转换为代码页 1051 和 1275

冰岛语

关于冰岛语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上冰岛语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------------|
| 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 和 Linux | 819, 923 |
| Apple 客户机 | 1275 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

871

不转换为代码页 858, 923, 924, 1275 和 5348

924

不转换为代码页 858 ,861 ,871 ,1051 ,1149 ,1252 ,1275 ,5348

1149

不转换为代码页 924,1051 和 1275

HP-UX

代码页:

1051

不转换为代码页 861

Windows

代码页:

861

不转换为代码页 1051 和 1275

东欧语言

东欧语言的 CCSID 和 CCSID 转换的详细信息。使用这些 CCSID 的典型语言包括阿尔巴尼亚语, 克罗地亚语, 捷克语, 匈牙利语, 波兰语, 罗马尼亚语, 塞尔维亚语, 斯洛伐克语和斯洛文尼亚语。

下表显示了受支持平台上东欧语言的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------------------|
| IBM i, z/OS | 870, 1153 |
| Windows | 852, 1250, 5346, 9044 |
| AIX, HP-UX, HP Integrity NonStop Server, Solaris 和 Linux | 912 |
| 东欧 Apple 客户端 | 1282 |
| 罗马尼亚语 Apple 客户端 | 1285 |
| 克罗地亚语 Apple 客户端 | 1284 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

z/OS

代码页:

870

不转换为代码页 1284,1285

1153

不转换为代码页 1250,1284 和 1285

IBM i

代码页:

870

不转换为代码页 1284,1285,5346 和 9044

1153

不转换为代码页 1282,1284,1285,5346 和 9044

HP-UX, Solaris 和 Linux

代码页:

912

不转换为代码页 1284,1285

HP Integrity NonStop Server

代码页:

912

不转换为代码页 1153,1284,1285,9044

Windows

代码页:

852

不转换为代码页 1284,1285

1250

不转换为代码页 1284,1285

9044

不转换为代码页 912,1282,1284,1285

西里尔文

西里尔文的 CCSID 和 CCSID 转换的详细信息。使用这些 CCSID 的典型语言包括白俄罗斯语，保加利亚语，马其顿语，俄语和塞尔维亚语。

下表显示了受支持平台上的西里尔文的本机 CCSID:

| 平台 | 本机 CCSID |
|--|----------------------------|
| 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 客户机 | 1283 |

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

IBM i

代码页:

880

不转换为代码页 855,866,878,1131,5347

1025

不转换为代码页 878 和 5347

Windows

代码页:

855

不转换为代码页 1131

866

不转换为代码页 1131

1131

不转换为代码页 855,866,880 和 1283

爱沙尼亚语

爱沙尼亚语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上爱沙尼亚语的本机 CCSID:

| 平台 | 本机 CCSID |
|-------------|------------|
| IBM i, z/OS | 1122, 1157 |

| 平台 | 本机 CCSID |
|-----------------------------|----------------------------|
| Windows | 902, 922, 1257, 5353, 9449 |
| AIX, HP-UX, Solaris 和 Linux | 902, 922 |
| HP Integrity NonStop Server | 922 |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

1122

不转换为代码页 902,1157 和 9449

1157

不转换为代码页 922,1122,1257 和 9449

IBM i

代码页:

1122

不转换为代码页 902,5353 和 9449

1157

不转换为代码页 922,5353 和 9449

HP-UX, Solaris 和 Linux

代码页:

902

不转换为代码页 922, 1122 和 9449

922

不转换为代码页 902,1157 和 9449

Windows

代码页:

5353

不转换为代码页 9449

9449

不转换为代码页 902,922,1122,1157,1257 和 5353

902

不转换为代码页 922, 1122 和 9449

HP Integrity NonStop Server

代码页:

922

不转换为代码页 902,1157 和 9449

拉脱维亚语和立陶宛语

拉脱维亚语和立陶宛语的 CCSID 和 CCSID 转换的详细信息。

下表显示了在受支持的平台上拉脱维亚语和立陶宛语的本机 CCSID:

| 平台 | 本机 CCSID |
|-----------------------------|----------------------------|
| IBM i, z/OS | 1112, 1156 |
| Windows | 901, 921, 1257, 5353, 9449 |
| AIX, HP-UX, Solaris 和 Linux | 901, 921 |
| HP Integrity NonStop Server | 921 |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

1112

不转换为代码页 901,1156 和 9449

1156

不转换为代码页 901,1156 和 9449

IBM i

代码页:

1112

不转换为代码页 5353

1153

不转换为代码页 921,5353 和 9449

HP-UX, Solaris 和 Linux

代码页:

902

不转换为代码页 921,1112,1257 和 9449

921

不转换为代码页 901,1156 和 9449

Windows

代码页:

901

不转换为代码页 921,1112,1257 和 9449

5355

不转换为代码页 9449

9449

不转换为代码页 901,921,1112,1156 和 1257

HP Integrity NonStop Server

代码页:

921

不转换为代码页 901,1156 和 9449

乌克兰语

乌克兰语 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上 Ukrainian 的本机 CCSID:

| 平台 | 本机 CCSID |
|--|------------------------|
| IBM i, z/OS | 1123 |
| Windows | 1124, 1125, 1251, 5347 |
| AIX, HP-UX, HP Integrity NonStop Server, Solaris 和 Linux | 1124 |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

IBM i

代码页:

1123

不转换为代码页 5347

HP-UX

代码页:

1124

不转换为代码页 5347

Windows

代码页:

1125

不转换为代码页 1123

希腊语

希腊语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上希腊语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------------|
| IBM i, z/OS | 875 |
| HP-UX | 813 (见附注) |
| Windows | 869, 1253, 5349 |
| AIX, NCR, HP Integrity NonStop Server, Solaris 和 Linux | 813 |
| Apple 客户机 | 1280 |
| DOS 客户机 | 737 |

注: 在 HP-UX 上仅支持 ISO 代码集。HP-UX 专有 greek8 代码集没有已注册的 CCSID 并且不受支持。

所有非客户机平台都支持在其本机 CCSID 和其他平台的本机 CCSID 之间进行转换，但有以下例外。

IBM i

代码页:

875

不转换为代码页 5349

Windows

代码页:

1253

不转换为代码页 737

5349

不转换为代码页 737

土耳其语

土耳其语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上土耳其语的本机 CCSID:

| 平台 | 本机 CCSID |
|---|-----------------|
| IBM i, z/OS | 1026 |
| HP-UX | 920 (见附注) |
| Windows | 857, 1254, 5350 |
| AIX, HP Integrity NonStop Server, Solaris 和 Linux | 920 |
| Apple 客户机 | 1281 |

注: 在 HP-UX 上仅支持 ISO 代码集。HP-UX 专有 turkish8 代码集没有已注册的 CCSID 并且不受支持。

所有非客户机平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

IBM i

代码页:

1026

不转换为代码页 5350

希伯来历

希伯来历的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上希伯来历的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------------------|
| z/OS | 424, 803, 4899, 12712 |
| IBM i | 424 |
| AIX | 916, 9048 |
| HP-UX | 916 (见附注) |
| Windows | 1255, 5351 |
| HP Integrity NonStop Server, Solaris 和 Linux | 916 |

注: 在 HP-UX 上仅支持 ISO 代码集。HP-UX 专有 greek8 代码集没有已注册的 CCSID 并且不受支持。

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

424

不转换为代码页 867,4899,9048,12712

803

不转换为代码页 867, 4899, 5351, 9048 和 12712

4899

不转换为代码页 424,803,856,862,916,1255

12712

不转换为代码页 424,803,856,916,1255

IBM i

代码页:

424

不转换为代码页 803,867, 4899, 5351, 9048 和 12712

代码页 424 也会转换为 CCSID 4952, 它是 856 的变体。

AIX

代码页:

916

不转换为代码页 867,4899,9048,12712

9048

不转换为代码页 424,803,856,862,916,1255

Windows

代码页:

1255

不转换为代码页 867,4899,9048,12712

5351

不转换为代码页 803

阿拉伯语

阿拉伯语的 CCSID 和 CCSID 转换的详细信息

下表显示了受支持平台上阿拉伯语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|----------------------|
| IBM i, z/OS | 420 |
| AIX | 1046, 1089 |
| HP-UX | 1089 (请参阅注释) |
| Windows | 720, 864, 1256, 5352 |
| HP Integrity NonStop Server, Solaris 和 Linux | 1089 |

注: 在 HP-UX 上仅支持 ISO 代码集。HP-UX 专有 arabic8 代码集没有已注册的 CCSID 并且不受支持。

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

420

不转换为代码页 5352

HP-UX, Solaris, Linux, HP Integrity NonStop Server, Tru64

代码页:

1089

不转换为代码页 720

Windows

代码页:

720

不转换为代码页 1089 和 5352

5352

不转换为代码页 720

波斯

Farsi 的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上 Farsi 的本机 CCSID:

| 平台 | 本机 CCSID |
|--|------------|
| IBM i, z/OS | 1097 |
| AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows | 1098 (见附注) |

注: 这些平台的本机 CCSID 尚未标准化, 可能发生更改。

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换。

乌尔都语

乌尔都语 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上 Urdu 的本机 CCSID:

| 平台 | 本机 CCSID |
|--|----------|
| IBM i, z/OS | 918 |
| Windows | 868 |
| AIX, HP-UX, HP Integrity NonStop Server, Solaris 和 Linux | 1006 |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

918

不转换为代码页 1006

泰国语

泰语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上泰语的本机 CCSID:

| 平台 | 本机 CCSID |
|-------------|----------|
| IBM i, z/OS | 838 |

| 平台 | 本机 CCSID |
|--|-----------|
| AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows | 874 (见附注) |
| 注: 这些平台的本机 CCSID 尚未标准化, 可能发生更改。 | |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换。

老挝语

老挝语的 CCSID 和 CCSID 转换的详细信息。

下表显示了老挝语在受支持平台上的本机 CCSID:

| 平台 | 本机 CCSID |
|--|----------|
| IBM i, z/OS | 1132 |
| AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows | 1133 |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换。

越南语

越南语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上越南语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|------------|
| IBM i, z/OS | 1130 |
| Windows | 1258, 5354 |
| AIX, HP-UX, HP Integrity NonStop Server, Solaris 和 Linux | 1129 |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

IBM i

代码页:

1130

不转换为代码页 1129 和 5354

日语拉丁语 SBCS

日语拉丁语 SBCS 的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上日语拉丁语 SBCS 的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------------------------|
| IBM i, z/OS | 1027 |
| AIX | 932, 5050 和 33722 (请参阅注释 1) |
| Windows | 932 和 943 (请参阅 Notes 2 和 3) |
| Linux, HP Integrity NonStop Server 和 Solaris | 943, 5050 |
| HP-UX | Not known |

| 平台 | 本机 CCSID |
|---|----------|
| <p>注:</p> <ol style="list-style-type: none"> 5050 和 33722 是与 AIX 上的基本代码页 954 相关的 CCSID。操作系统报告的 CCSID 为 33722。 Windows NT 使用代码页 932，但这最好由 CCSID 943 表示。但是，并非所有 WebSphere MQ 平台都支持此 CCSID。 在 WebSphere MQ for Windows CCSID 932 用于表示代码页 932，但可以更改文件 ../conv/table/ccsid.tbl 以将 CCSID 更改为 943。 WebSphere MQ 不支持基于 JIS X 0213 (JIS2004) 标准的代码页。 | |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

1027

不转换为代码页 932,942,943,954,5050 和 33722

IBM i

代码页:

1027

不转换为代码页 932

AIX

代码页:

932

不转换为代码页 1027

5050

不转换为代码页 1027

33722

不转换为代码页 1027

Linux

代码页:

943

不转换为代码页 1027

5050

不转换为代码页 1027

Solaris

代码页:

943

不转换为代码页 1027

5050

不转换为代码页 1027

HP Integrity NonStop Server

代码页:

943

不转换为代码页 1027

5050

不转换为代码页 1027

日语片假名 SBCS

日语片假名 SBCS 的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上日语片假名 SBCS 的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------------------------|
| IBM i, z/OS | 290 |
| HP-UX | 897 |
| AIX | 932 ,5050 和 33722 (请参阅注释 1) |
| Windows | 932 和 943 (请参阅 Notes 2 和 3) |
| Linux, HP Integrity NonStop Server 和 Solaris | 943, 5050 |

注:

- 5050 和 33722 是与 AIX 上的基本代码页 954 相关的 CCSID。操作系统报告的 CCSID 为 33722。
- Windows NT 使用代码页 932，但这最好由 CCSID 943 表示。但是，并非所有 WebSphere MQ 平台都支持此 CCSID。
在 WebSphere MQ for Windows CCSID 932 用于表示代码页 932，但可以更改文件 ../conv/table/ccsid.tbl 以将 CCSID 更改为 943。
- WebSphere MQ 不支持基于 JIS X 0213 (JIS2004) 标准的代码页。
- 除上述转换外，AIX，HP-UX，Solaris Linux 和 Tru64 上的 WebSphere MQ 产品支持从 CCSID 897 到 CCSID 37,273，277，278，280，284，285，290，297，437，500，819，850，1027 和 1252 的转换。

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

290

不转换为代码页 932 ,943 ,954 ,5050 和 33722

IBM i

代码页:

290

不转换为代码页 932

AIX

代码页:

932

不转换为代码页 290 和 897

5050

不转换为代码页 290 和 897

33722

不转换为代码页 290 和 897

HP-UX

代码页:

897

不转换为代码页 932 ,943 ,954 ,5050 和 33722

Linux

代码页:

943

不转换为代码页 290 和 897

5050

不转换为代码页 290 和 897

Solaris

代码页:

943

不转换为代码页 290 和 897

5050

不转换为代码页 290 和 897

HP Integrity NonStop Server

代码页:

943

不转换为代码页 290 和 897

5050

不转换为代码页 290 和 897

日语日语汉字/拉丁语混合

日语日语汉字/拉丁语混合的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上日语日语汉字/拉丁语混合的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------------------------|
| IBM i, z/OS | 1399,5035 (请参阅注释 1) |
| AIX | 932 ,5050 和 33722 (请参阅注释 2) |
| HP-UX | 932 ,954 和 5039 (请参阅注释 3) |
| Windows | 932 和 943 (请参阅 Notes 4 和 5) |
| Linux, HP Integrity NonStop Server 和 Solaris | 943, 5050 |

| 平台 | 本机 CCSID |
|---|----------|
| <p>注:</p> <ol style="list-style-type: none"> 5035 是与代码页 939 相关的 CCSID 5050 和 33722 是与 AIX 上的基本代码页 954 相关的 CCSID。操作系统报告的 CCSID 为 33722。 HP-UX 上的代码集 japan15 和 SJIS 由 CCSID 932 表示。这些字符在 SJIS 中有几个具有不同表示的 DBCS 字符，因此如果未在 HP-UX 系统上执行转换，那么 932 可能无法正确转换。WebSphere MQ for HP-UX 支持 5039，这是 HP SJIS 的正确 CCSID。可以对文件 /var/mqm/conv/ccsid.tbl 进行更改，以将使用的 CCSID 从 932 更改为 5039。 Windows NT 使用代码页 932，但这最好由 CCSID 943 表示。但是，并非所有 WebSphere MQ 平台都支持此 CCSID。 在 WebSphere MQ for Windows CCSID 932 用于表示代码页 932，但可以更改文件 ../conv/table/ccsid.tbl 以将 CCSID 更改为 943。 WebSphere MQ 不支持基于 JIS X 0213 (JIS2004) 标准的代码页。 | |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

1399

不转换为代码页 954,5035,5050,33722

5035

不转换为代码页 954， 1399， 5050 和 33722

IBM i

代码页:

1399

不转换为代码页 5039

5035

不转换为代码页 5039

HP-UX

代码页:

932

不转换为代码页 942， 943 和 1399

954

不转换为代码页 942， 943 和 1399

5039

不转换为代码页 942， 943 和 1399

HP Integrity NonStop Server

代码页:

943

不转换为代码页 1399

5050

不转换为代码页 1399

日语日语汉字/片假名混合

日语日语汉字/片假名混合的 CCSID 和 CCSID 转换的详细信息。

| 平台 | 本机 CCSID |
|--|-----------------------------|
| z/OS | 1390,5026 (请参阅注释 1) |
| IBM i | 5026 (请参阅注释 1) |
| AIX | 932 ,5050 和 33722 (请参阅注释 2) |
| HP-UX | 932 ,954 和 5039 (请参阅注释 3) |
| Windows | 932 和 943 (请参阅 Notes 4 和 5) |
| Linux, HP Integrity NonStop Server 和 Solaris | 943, 5050 |

注:

1. CCSID 1390 不接受小写字符。5026 是与代码页 930 相关的 CCSID。当选择日语片假名 (DBCS) 功能部件时, CCSID 5026 是在 IBM i 上报告的 CCSID。
2. 5050 和 33722 是与 AIX 上的基本代码页 954 相关的 CCSID。操作系统报告的 CCSID 为 33722。
3. HP-UX 上的代码集 japan15 和 SJIS 由 CCSID 932 表示。这些字符在 SJIS 中有几个具有不同表示的 DBCS 字符, 因此如果未在 HP-UX 系统上执行转换, 那么 932 可能无法正确转换。WebSphere MQ for HP-UX 支持 5039, 这是 HP SJIS 的正确 CCSID。可以对文件 `/var/mqm/conv/ccsid.tbl` 进行更改, 以将使用的 CCSID 从 932 更改为 5039。
4. Windows NT 使用代码页 932, 但这最好由 CCSID 943 表示。但是, 并非所有 WebSphere MQ 平台都支持此 CCSID。
在 WebSphere MQ for Windows 上, CCSID 932 用于表示代码页 932, 但可以更改文件 `../conv/table/ccsid.tbl` 以将 CCSID 更改为 943。
5. WebSphere MQ 不支持基于 JIS X 0213 (JIS2004) 标准的代码页。

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

z/OS

代码页:

1390

不转换为代码页 954,5026,5050,33722

不接受小写字符。

5026

不转换为代码页 954, 1390, 5050 和 33722

IBM i

代码页:

5026

不转换为代码页 1390 和 5039

HP-UX

代码页:

932

不转换为代码页 942,943 和 1390

954

不转换为代码页 942,943 和 1390

5039

不转换为代码页 942,943 和 1390

HP Integrity NonStop Server

代码页:

943

不转换为代码页 1390

5050

不转换为代码页 1390

韩语

韩语的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上韩语的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------|
| z/OS, IBM i | 933, 1364 |
| AIX, HP-UX, Linux, HP Integrity NonStop Server 和 Solaris | 970 |
| Windows | 949, 1363 |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换, 但以下情况例外。

z/OS

代码页:

933

不转换为代码页 970

1364

不转换为代码页 970

HP-UX

代码页:

970

不转换为代码页 999,1363,1364

简体中文

简体中文的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上的简体中文的本机 CCSID:

| 平台 | 本机 CCSID |
|--|---------------------|
| z/OS | 935, 1388 |
| IBM i | 935, 1388 |
| AIX | 1383, 1386 |
| HP-UX | 1381 (请参阅注释 1) |
| Windows | 1381,1386 (请参阅注释 2) |
| Linux, HP Integrity NonStop Server 和 Solaris | 1383 |

| 平台 | 本机 CCSID |
|--|----------|
| <p>注:</p> <ol style="list-style-type: none"> HP-UX 上的代码集 prc15 和 hp15CN 由 CCSID 1381 表示。 Windows 使用代码页 936，但这最好由 CCSID 1386 表示。但是，并非所有 WebSphere MQ 平台都支持此 CCSID。 在 WebSphere MQ for Windows CCSID 1381 用于表示代码页 936，但是可以对文件 ../conv/table/ccsid.tbl 进行更改，从而将使用的 CCSID 更改为 1386。 WebSphere MQ 支持中文 GB18030 标准的第一阶段。 在 z/OS, Linux, Windows 和 Solaris 上，提供了 Unicode (UTF-8 和 UCS-2) 与 CCSID 1388 (具有 GB18030 扩展名的 EBCDIC)，Unicode (UTF-8 和 UCS-2) 与 CCSID 5488 (GB18030 第一阶段) 之间以及 CCSID 1388 与 CCSID 5488 之间的转换支持。 注: 在 IBM i 上，操作系统支持 Unicode (UTF-8 和 UCS-2) 与 CCSID 1388 (具有 GB18030 扩展名的 EBCDIC) 之间的转换。 在 HP-UX 上，HP11 操作系统上当前不支持 GB18030。在 HP11i 上，补丁 PHCO_26456 提供 GB18030 (CCSID 5488) 与 Unicode 之间的转换支持。不支持在 GB18030 与 1388 (EBCDIC) 之间进行转换。 | |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

935

不转换为代码页 1383

1388

不转换为代码页 1383

HP-UX

代码页:

1381

不转换为代码页 1383,1386 和 1388

繁体中文

繁体中文的 CCSID 和 CCSID 转换的详细信息。

下表显示了受支持平台上的繁体中文的本机 CCSID:

| 平台 | 本机 CCSID |
|--|-----------------------|
| z/OS, IBM i | 937 |
| HP-UX | 938,950 和 964 (请参阅注释) |
| Windows | 950 |
| AIX, HP Integrity NonStop Server, Solaris 和 Linux | 950, 964 |
| <p>注: HP-UX 上的代码集 roc15 由 CCSID 938 表示。</p> | |

所有平台都支持在其本机 CCSID 与其他平台的本机 CCSID 之间进行转换，但以下情况例外。

z/OS

代码页:

937

不转换为代码页 964

1388

不转换为代码页 1383

HP-UX

代码页:

938

不转换为代码页 948

950

不转换为代码页 948

964

不转换为代码页 948

Linux 和 Solaris

代码页:

964

不转换为代码页 938

Unicode 转换支持

某些平台支持将用户数据转换为 Unicode 编码或从 Unicode 编码转换为用户数据。支持两种形式的 Unicode 编码: UCS-2 (CCSID 1200,13488 和 17584) 和 UTF-8 (CCSID 1208)。

术语 *UCS-2* 通常可互换使用, 但与 *UTF-16* 不正确使用。UCS-2 是固定宽度的编码, 其中每个字符占用 2 个字节。UTF-16 是变量宽度编码, 它是 UCS-2 的超集。除了 2-byte UCS-2 字符外, UTF-16 还包含长度为 4 字节的字符 (称为代理对)。WebSphere MQ 不支持代理对。因此, WebSphere MQ 中对 UTF-16 和 UTF-8 的支持仅限于可以在 UCS-2 中编码的 Unicode 字符。

注: WebSphere MQ 不支持 UCS-2 队列管理器 CCSID, 因此无法在 UCS-2 中对消息头数据进行编码。

WebSphere MQ AIX 支持 Unicode

在 WebSphere MQ for AIX 上, 下表中的 CCSID 支持与 Unicode CCSID 之间的转换。

| | | | | | |
|------|------|------|------|------|------|
| 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 支持 Unicode

在 WebSphere MQ for HP-UX 上，下表中列出的 CCSID 支持与 Unicode CCSID 之间的转换。

| | | | | | |
|------|------|-------|-------|------|------|
| 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 for Windows, Solaris 和 Linux 支持 Unicode

在 WebSphere MQ for Windows, WebSphere MQ for Solaris 以及 WebSphere MQ for Linux 上，下表中的 CCSID 支持从 Unicode CCSID 转换为 Unicode CCSID。

| | | | | | |
|------|---------|---------|---------|------|---------|
| 037 | 277 | 278 | 280 | 284 | 285 |
| 290 | 297 | 300 | 301 | 420 | 424 |
| 437 | 500 | 813 | 819 | 833 | 835 |
| 836 | 837 | 838 | 850 | 852 | 855 |
| 856 | 857 | 858 | 860 | 861 | 862 |
| 863 | 864 | 865 | 866 | 867 | 868 |
| 869 | 870 | 871 | 874 | 875 | 878 |
| 880 | 891 | 897 | 901 | 902 | 903 |
| 904 | 912 | 913 (5) | 915 | 916 | 918 |
| 920 | 921 | 922 | 923 | 924 | 927 |
| 928 | 930 | 931 (1) | 932 (2) | 933 | 935 |
| 937 | 938 (3) | 939 | 941 | 942 | 943 |
| 947 | 948 | 949 | 950 | 951 | 954 (4) |
| 964 | 970 | 1006 | 1025 | 1026 | 1027 |
| 1040 | 1041 | 1042 | 1043 | 1046 | 1047 |
| 1051 | 1088 | 1089 | 1097 | 1098 | 1112 |

| | | | | | |
|-------|-------|----------|-----------|------|------|
| 1114 | 1115 | 1122 | 1123 | 1124 | 1129 |
| 1130 | 1132 | 1133 | 1140 | 1141 | 1142 |
| 1143 | 1144 | 1145 | 1146 | 1147 | 1148 |
| 1149 | 1153 | 1156 | 1157 | 1200 | 1208 |
| 1250 | 1251 | 1252 | 1253 | 1254 | 1255 |
| 1256 | 1257 | 1258 | 1275 | 1280 | 1281 |
| 1282 | 1283 | 1363 | 1364 | 1380 | 1381 |
| 1383 | 1386 | 1388 | 4899 | 5050 | 5346 |
| 5347 | 5348 | 5349 | 5350 | 5351 | 5352 |
| 5353 | 5354 | 5488 (5) | 9044 | 9048 | 9449 |
| 12712 | 13488 | 17584 | 33722 (4) | | |

注意:

1. 931 使用 939 进行转换。
2. 932 使用 942 进行转换。
3. 938 使用 948 进行转换。
4. 954 和 33722 使用 5050 进行转换。
5. 仅在 Windows, Linux 和 Solaris 上。

IBM i 支持 Unicode

有关 UNICODE 支持的详细信息，请参阅与您的操作系统相关的相应 IBM i 出版物。

WebSphere MQ for z/OS 支持 Unicode

在 WebSphere MQ for z/OS 上，以下 CCSID 支持与 Unicode CCSID 之间的转换:

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| 37 | 256 | 259 | 273 | 275 | 277 |
| 278 | 280 | 282 | 284 | 285 | 290 |
| 293 | 297 | 300 | 301 | 367 | 420 |
| 423 | 424 | 437 | 500 | 720 | 737 |
| 775 | 803 | 806 | 808 | 813 | 819 |
| 833 | 834 | 835 | 836 | 837 | 838 |
| 848 | 849 | 850 | 851 | 852 | 855 |
| 856 | 857 | 858 | 859 | 860 | 861 |
| 862 | 863 | 864 | 865 | 866 | 867 |
| 868 | 869 | 870 | 871 | 872 | 874 |
| 875 | 878 | 880 | 891 | 895 | 896 |
| 897 | 901 | 902 | 903 | 904 | 905 |
| 912 | 914 | 915 | 916 | 918 | 920 |
| 921 | 922 | 923 | 924 | 927 | 928 |
| 930 | 932 | 933 | 935 | 937 | 939 |
| 941 | 942 | 943 | 944 | 946 | 947 |

| | | | | | |
|-------|-------|-------|-------|-------|-------|
| 948 | 949 | 950 | 951 | 1004 | 1006 |
| 1008 | 1009 | 1010 | 1011 | 1012 | 1013 |
| 1014 | 1015 | 1016 | 1017 | 1018 | 1019 |
| 1025 | 1026 | 1027 | 1040 | 1041 | 1042 |
| 1043 | 1046 | 1047 | 1051 | 1088 | 1089 |
| 1097 | 1098 | 1112 | 1114 | 1115 | 1122 |
| 1123 | 1124 | 1125 | 1126 | 1129 | 1130 |
| 1131 | 1132 | 1133 | 1137 | 1140 | 1141 |
| 1142 | 1143 | 1144 | 1145 | 1146 | 1147 |
| 1148 | 1149 | 1153 | 1154 | 1155 | 1156 |
| 1157 | 1158 | 1159 | 1160 | 1161 | 1162 |
| 1164 | 1200 | 1208 | 1250 | 1251 | 1252 |
| 1253 | 1254 | 1255 | 1256 | 1257 | 1258 |
| 1275 | 1276 | 1277 | 1280 | 1281 | 1282 |
| 1283 | 1284 | 1285 | 1351 | 1362 | 1363 |
| 1364 | 1370 | 1371 | 1380 | 1381 | 1385 |
| 1386 | 1388 | 1390 | 1399 | 4899 | 4909 |
| 4930 | 4933 | 4948 | 4951 | 4952 | 4960 |
| 4971 | 5012 | 5039 | 5104 | 5123 | 5142 |
| 5210 | 5346 | 5347 | 5348 | 5349 | 5350 |
| 5351 | 5352 | 5353 | 5354 | 5488 | 8482 |
| 8612 | 9027 | 9030 | 9044 | 9048 | 9049 |
| 9056 | 9061 | 9066 | 9238 | 9449 | 12712 |
| 13121 | 13218 | 13488 | 16684 | 16804 | 17248 |
| 17584 | 21427 | 28709 | | | |

64 位平台上的编码标准

使用此信息可了解有关 64 位平台上的编码标准以及首选数据类型的信息。

首选数据类型

这些类型从不更改大小，并且在 32 位和 64 位 WebSphere MQ 平台上都可用：

| 名称 | 长度 |
|----------|------|
| MQLONG | 4 字节 |
| MQULONG | 4 字节 |
| MQINT32 | 4 字节 |
| MQUINT32 | 4 字节 |
| MQINT64 | 8 字节 |
| MQUINT64 | 8 字节 |

标准数据类型

了解 32 位 UNIX, 64 位 UNIX 和 64 位 Windows 应用程序上的标准数据类型。

32 位 UNIX 应用程序

包含此部分以进行比较, 并基于 Solaris。注意与其他 UNIX 平台的任何差异:

| 名称 | 长度 |
|-------------|--|
| char | 1 个字节 |
| 短整型 | 2 字节 |
| int | 4 字节 |
| long | 4 字节 |
| 浮点值 | 4 字节 |
| 双精度值 | 8 字节 |
| long double | 16 字节 |
| | 请注意, 在 AIX 和 Linux PPC 上, 长双精度值为 8 字节。 |
| 指针 | 4 字节 |
| ptrdiff_t | 4 字节 |
| 大小 | 4 字节 |
| 时间 (t) | 4 字节 |
| 发条 | 4 字节 |
| wchar_t | 4 字节 |
| | 请注意, 在 AIX 上, wchar_t 为 2 字节。 |

64 位 UNIX 应用程序

本部分基于 Solaris。注意与其他 UNIX 平台的任何差异:

| 名称 | 长度 |
|-------------|--|
| char | 1 个字节 |
| 短整型 | 2 字节 |
| int | 4 字节 |
| long | 8 字节 |
| 浮点值 | 4 字节 |
| 双精度值 | 8 字节 |
| long double | 16 字节 |
| | 请注意, 在 AIX 和 Linux PPC 上, 长双精度值为 8 字节。 |
| 指针 | 8 字节 |
| ptrdiff_t | 8 字节 |
| 大小 | 8 字节 |
| 时间 (t) | 8 字节 |

| 名称 | 长度 |
|---------|--|
| 发条 | 8 字节 请注意，在其他 UNIX 平台上，clock_t 为 4 字节。 |
| wchar_t | 4 字节 请注意，在 AIX 上，wchar_t 为 2 字节。 |

Windows 64 位应用程序

| 名称 | 长度 |
|-------------|--------------------------|
| char | 1 个字节 |
| 短整型 | 2 字节 |
| int | 4 字节 |
| long | 4 字节 |
| 浮点值 | 4 字节 |
| 双精度值 | 8 字节 |
| long double | 8 字节 |
| 指针 | 8 字节 请注意，所有指针都是 8 字节。 |
| ptrdiff_t | 8 字节 |
| 大小 | 8 字节 |
| 时间 (t) | 8 字节 |
| 发条 | 4 字节 |
| wchar_t | 2 字节 |
| WORD | 2 字节 |
| DWORD | 4 字节 |
| 句柄 | 8 字节 |
| HFILE | 4 字节 |

Windows 上的编码注意事项

HANDLE hf;

适用平台

```
hf = CreateFile((LPCTSTR) FileName,
               Access,
               ShareMode,
               xihSecAttsNTRestrict,
               Create,
               AttrAndFlags,
               NULL);
```

请勿使用

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                       Access,
```



```
ShareMode,  
xihSecAttsNTRestrict,  
Create,  
AttrAndFlags,  
NULL);
```

因为这会产生错误。

size_t len fgets

适用平台

```
size_t len  
while (fgets(string1, (int) len, fp) != NULL)  
len = strlen(buffer);
```

请勿使用

```
int len;  
while (fgets(string1, len, fp) != NULL)  
len = strlen(buffer);
```

printf

适用平台

```
printf("My struc pointer: %p", pMyStruc);
```

请勿使用

```
printf("My struc pointer: %x", pMyStruc);
```

如果需要十六进制输出，那么必须分别打印 4 字节的上限和下限。

char * 指针

适用平台

```
char * ptr1;  
char * ptr2;  
size_t bufLen;  
  
bufLen = ptr2 - ptr1;
```

请勿使用

```
char *ptr1;  
char *ptr2;  
UINT32 bufLen;  
  
bufLen = ptr2 - ptr1;
```

alignBytes

适用平台

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

请勿使用

```
void *address;  
unsigned short alignBytes;  
  
alignBytes = (unsigned short) ((UINT32) address % 16);
```

len

适用平台

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

请勿使用

```
void *address1;  
void *address2;  
UINT32 len;  
  
len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf

适用平台

```
MQLONG SBCSprt;  
sscanf(line, "%d", &SBCSprt);
```

请勿使用

```
MQLONG SBCSprt;  
sscanf(line, "%1d", &SBCSprt);
```

%1d 尝试将 8 字节类型放入 4 字节类型中; 仅当您处理的是实际 long 数据类型时才使用 %1。MQLONG, UINT32 和 INT32 定义为四个字节, 与所有 WebSphere MQ 平台上的 int 相同:

SOAP 引用

按字母顺序排列的 WebSphere MQ Transport for SOAP 参考信息。

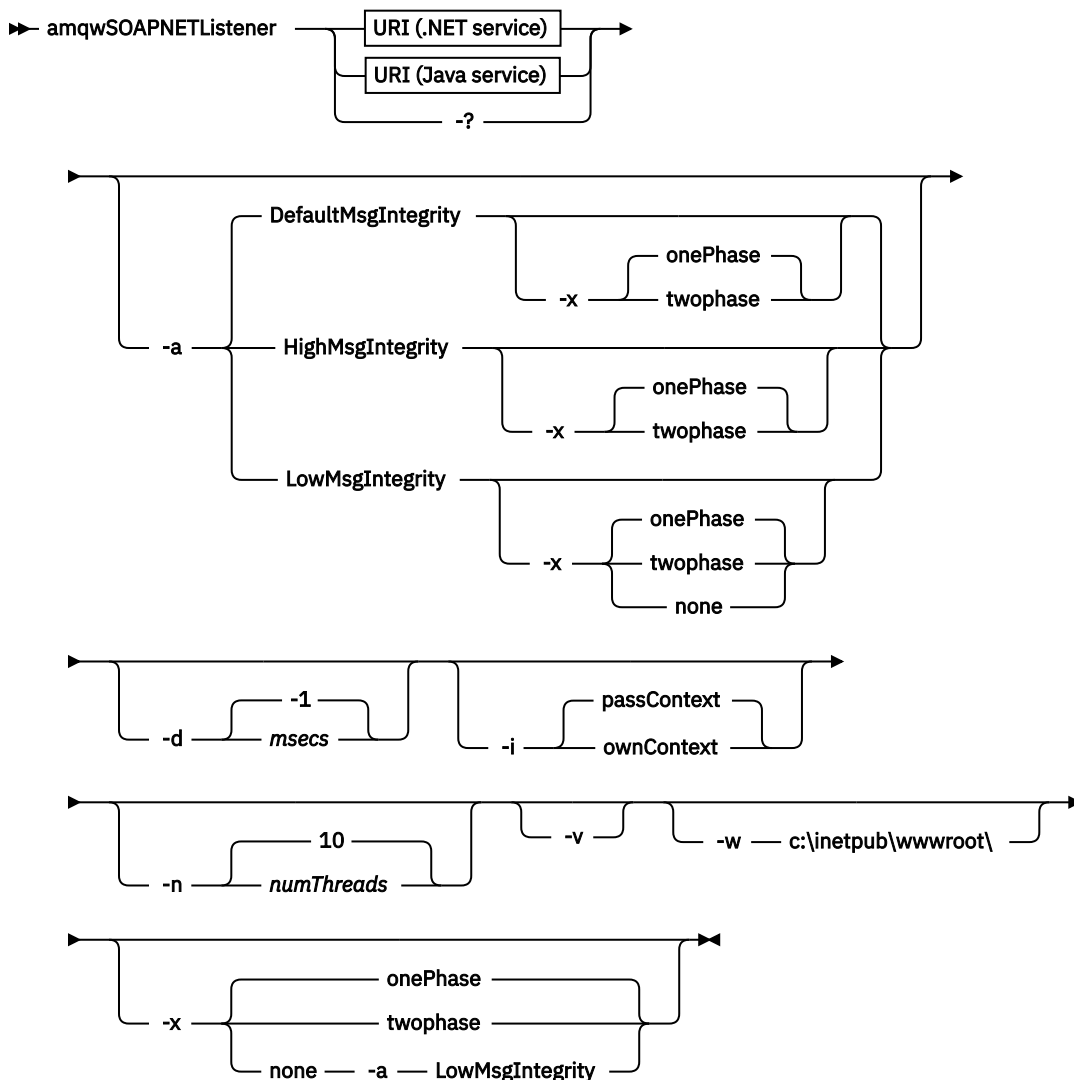
amqwSOAPNETListener: IBM WebSphere MQ SOAP 侦听器 for .NET Framework 1 或 2

WebSphere MQ SOAP 侦听器 for .NET Framework 1 或 2 的语法和参数。

用途

启动 IBM WebSphere MQ SOAP 侦听器 for .NET Framework 1 或 2。

.NET



必需参数

URI 平台

请参阅第 885 页的『Web Service 部署的 URI 语法和参数』。

-?

打印描述如何使用命令的帮助文本。

可选参数

-a integrityOption

integrityOption 指定 WebSphere MQ SOAP 侦听器的行为 (如果无法将失败请求消息放入死信队列)。
integrityOption 可以采用下列其中一个值:

DefaultMsgIntegrity

对于非持久消息, 侦听器将显示一条警告消息, 并在废弃原始消息的情况下继续执行。对于持久消息, 它会显示错误消息, 回退请求消息以使其保留在请求队列上并退出。如果省略了 -a 选项, 或者未指定 *integrityOption*, 那么 DefaultMsgIntegrity 适用。

LowMsgIntegrity

对于持久消息和非持久消息, 侦听器会显示警告并继续执行, 从而废弃该消息。

HighMsgIntegrity

对于持久消息和非持久消息, 侦听器会显示一条错误消息, 回退请求消息以使其保留在请求队列中并退出。

部署实用程序检查 **-x** 和 **-a** 标志的兼容性。如果指定了 **-x none**，那么必须指定 **-a LowMsgIntegrity**。如果标志不兼容，那么部署实用程序将退出并返回错误消息，并且未执行任何部署步骤。

-d 毫秒

msecs 指定在任何线程上接收到请求消息时，WebSphere MQ SOAP 侦听器保持活动的毫秒数。如果 *msecs* 设置为 **-1**，那么侦听器将无限期保持活动状态。

-i 上下文

上下文 指定侦听器是否传递身份上下文。上下文 采用以下值：

passContext

将原始请求消息的身份上下文设置到响应消息中。SOAP 侦听器检查其是否有权从请求队列保存上下文并将其传递到响应队列。当打开请求队列以保存上下文和响应队列以传递上下文时，它会在运行时进行检查。如果它没有必需的权限，或者 **MQOPEN** 调用失败，并且未处理响应消息。响应消息将与包含来自失败 **MQOPEN** 的返回码的死信头一起放在死信队列上。然后，侦听器继续正常处理后续入局消息。

ownContext

SOAP 侦听器未传递上下文。返回的上下文反映用于运行侦听器的用户标识，而不是用于创建原始请求消息的用户标识。

源上下文中的字段由队列管理器设置，而不是由 SOAP 侦听器设置。

-n numThreads

numThreads 指定为 WebSphere MQ SOAP 侦听器生成的启动脚本中的线程数。缺省值为 **10**。如果消息吞吐量较高，请考虑增大此数字。

-v

-v 设置来自外部命令的详细输出。始终显示错误消息。使用 **-v** 输出可定制以创建定制部署脚本的命令。

-w serviceDirectory

serviceDirectory 是包含 Web Service 的目录。

-x 事务性

事务性 指定侦听器的事务控制类型。可以将 事务性 设置为下列其中一个值：

onePhase

使用 IBM WebSphere MQ 一阶段支持。如果系统在处理期间失败，那么会将请求消息重新传递到应用程序。WebSphere MQ 事务确保响应消息只写入一次。

twoPhase

使用两阶段支持。如果适当地编写服务，那么将在服务的单个已落实执行中，与其他资源协调，仅传递一次消息。此选项仅适用于服务器绑定连接。

none

无事务支持。如果系统在处理期间失败，那么请求消息可能会丢失，即使是持久消息也是如此。该服务可能已执行，也可能未执行，并且可能已写入或未写入响应，报告或死信消息。

部署实用程序检查 **-x** 和 **-a** 标志的兼容性。请参阅 **-a** 标志的描述以获取详细信息。

.NET 示例

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.Nojndi"
-w C:/wmqsoap/demos
-n 20
```

amqswsd1: 为 .NET Framework 1 或 2 服务生成 WSDL

amqswsd1 采用针对 .NET Framework 1 或 2 编写的 Web Service，并生成该类的 WSDL，将您为 WebSphere MQ Transport for SOAP 提供的 URI 插入到生成的 WSDL 中。

用途

使用 **amqswsd1** 生成包含部署到 WebSphere MQ 的服务的 URI 的 WSDL。使用 WSDL 来生成客户机代理。

►► amqswsdl — *escapedUri* — *className* — .asmx — *className* — .wsdl ◀◀

参数

escapedUri (输入)

服务的 URI，将所有“&”转义为“&”。例如：

```
"jms:/queue?destination=REQUESTDOTNET
&amp.initialContextFactory=com.ibm.mq.jms.Nojndi
&amp.connectionFactory=(connectQueueManager(QM1)binding(server))
&amp.targetService=Quote.asmx"
```

className.asmx (输入)

服务类。

className.wsdl (输出)

服务 WSDL。

描述

如果类是使用后代编程模型实现的，那么必须构建 *className.dll* 并将其存储在 *./bin* 中。

amqwsdlclientconfig: 为 WebSphere MQ Transport for SOAP 创建 Axis 1.4 Web Service 客户机部署描述符

amqwsdlclientconfig 将创建 *client-config.wsdd* Axis 1.4 客户机部署描述符文件。

用途

它将 *jms:/* 传输添加到描述符，并将 *java:com.ibm.mq.soap.transport.jms.WMQSender* 注册为用于处理 *jms:* 传输的 SOAP 请求的类。

语法

►► amqwsdlclientconfig ◀◀

描述

amqwsdlclientconfig 调用 **amqwsdlsetcp** 以设置 CLASSPATH 并运行命令：

```
java org.apache.axis.utils.Admin client "%WMSOAP_HOME%\bin\amqwsdlclientTransport.wsdd"
```

amqwsdldeployWMQService: 部署 Web Service 实用程序

部署实用程序准备服务类以用作将 WebSphere MQ 用作传输的 Web Service。

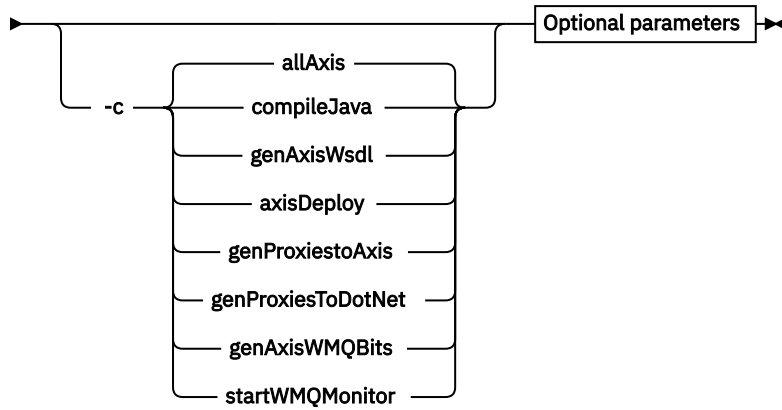
用途

使用部署实用程序来生成部署 Axis 1.4，.NET Framework 1 或 .NET Framework 2 服务所需的文件。使用这些文件来部署由 IBM WebSphere MQ 调用的服务。**amqwsdldeployWMQService** 生成的文件显示在 [第 858 页的『来自 amqwsdldeployWMQService 的输出文件』](#) 中。

Syntax diagram

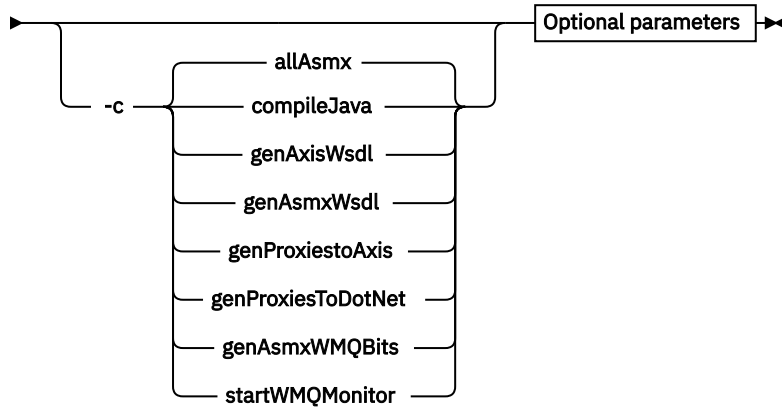
UNIX and Linux systems

▶▶ ./amqwdeployWMQService.sh -f *className* -?

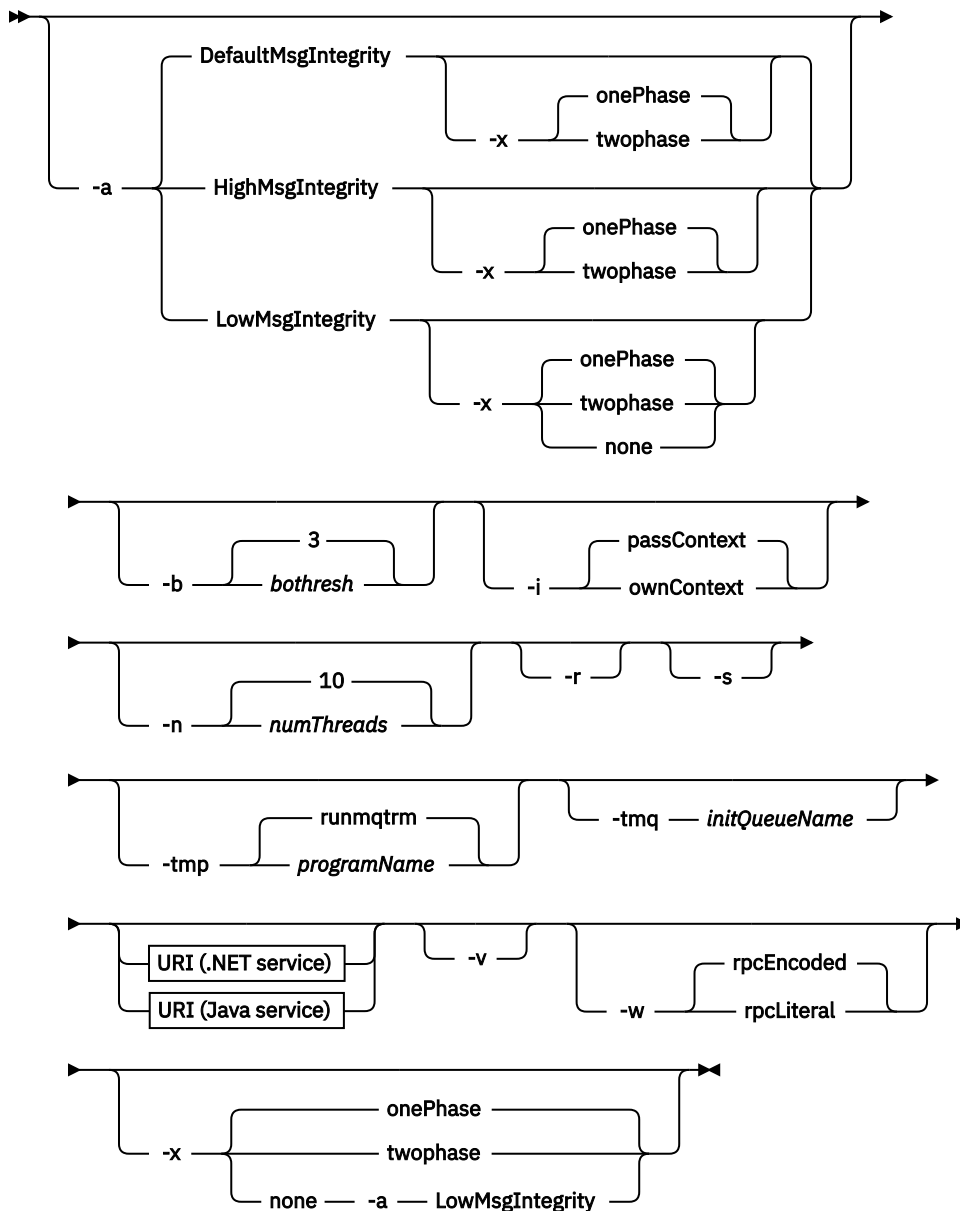


Windows

▶▶ amqwdeployWMQService -f *className* -?



Optional parameters



必需参数

-f *className*

className 是要部署的类的名称。对于 Axis 服务 *className* 是 Java 源文件，而对于 .NET 服务，是 .asmx 文件。第 855 页的图 11 说明了 .NET 服务的 Axis 服务和第 855 页的图 12 的部署。

```
amqwdeployWmqService -f javaDemos/service/StockQuoteAxis.java
```

图 11: Axis 服务的示例部署

```
amqwdeployWmqService -f StockQuoteDotNet.asmx
```

图 12: .NET 服务的示例部署

对于 Java，*className* 必须由包名进行标准限定。可以将其指定为具有目录分隔符的路径名或具有句点分隔符的类名。生成的类位于 `./generated/client/remote/path name` 中。对于 .NET 服务，虽然可以指定目录，但生成的 Java 代理始终位于 `./generated/client/remote/dotNetService` 中。

如果使用 `-u` 选项指定 URI 并在 URI 中指定 `targetService`，那么部署实用程序将检查 `className`。`className` 必须与 `targetService` 匹配。如果类和服务不匹配，那么部署实用程序将显示错误消息并退出。

-?

打印描述如何使用命令的帮助文本。

可选参数

-a *integrityOption*

integrityOption 指定 WebSphere MQ SOAP 侦听器的行为 (如果无法将失败的请求消息放入死信队列)。*integrityOption* 可以采用下列其中一个值:

DefaultMsgIntegrity

对于非持久消息，侦听器将显示一条警告消息，并在废弃原始消息的情况下继续执行。对于持久消息，它会显示错误消息，回退请求消息以使其保留在请求队列上并退出。如果省略了 `-a` 选项，或者未指定 *integrityOption*，那么 `DefaultMsgIntegrity` 适用。

LowMsgIntegrity

对于持久消息和非持久消息，侦听器会显示警告并继续执行，从而废弃该消息。

HighMsgIntegrity

对于持久消息和非持久消息，侦听器会显示一条错误消息，回退请求消息以使其保留在请求队列中并退出。

部署实用程序检查 `-x` 和 `-a` 标志的兼容性。如果指定了 `-x none`，那么必须指定 `-a LowMsgIntegrity`。如果标志不兼容，那么部署实用程序将退出并返回错误消息，并且未执行任何部署步骤。

-b *bothresh*

bothresh 指定请求队列的回退阈值设置。缺省值为 3。

-c 操作

operation 指定要执行的部署过程的哪个部分。*operation* 是下列其中一个选项:

allAxis

对 Axis 或 Java 服务执行所有编译和设置步骤⁴。

compileJava

将 Java 服务 `.java` 编译为 `.class`。

genAxisWsd1

生成 WSDL: `.class` 到 `.wsdl`。

axisDeploy

将类文件 `.wsdl` 部署到 `.wsdd`，应用 `.wsdd`。

genProxiestoAxis

生成代理: `.wsdl` 到 `.java` 和 `.class`。

genAxisWMQBits

为 Axis 服务设置 IBM WebSphere MQ 队列，IBM WebSphere MQ SOAP 侦听器 and 触发器。

allAsmx

对 .NET 服务执行所有设置步骤⁵。

genAsmxWsd1

生成 WSDL: `.asmx` 到 `.wsdl`。

genProxiesToDotNet

生成代理: `.wsdl` 到 `.java`，`.class`，`.cs` 和 `.vb`。

genAsmxWMQBits

设置 IBM WebSphere MQ 队列，IBM WebSphere MQ SOAP 侦听器 and 触发器

startWMQMonitor

启动 WebSphere MQ SOAP 服务的触发器监视器。

⁴ 缺省值 (如果 `className` 具有 `.java` 扩展名)

⁵ 缺省值 (如果 `className` 具有 `.asmx` 扩展名)。

注: `runmqtrm` 以 `mqm` 用户标识运行。如果安全性是问题, 那么必须确保在相应的用户标识下启动侦听器。

-i 上下文

上下文 指定侦听器是否传递身份上下文。上下文 采用以下值:

passContext

将原始请求消息的身份上下文设置到响应消息中。SOAP 侦听器检查其是否有权从请求队列保存上下文并将其传递到响应队列。当打开请求队列以保存上下文和响应队列以传递上下文时, 它会在运行时进行检查。如果它没有必需的权限, 或者 `MQOPEN` 调用失败, 并且未处理响应消息。响应消息将与包含来自失败 `MQOPEN` 的返回码的死信头一起放在死信队列上。然后, 侦听器继续正常处理后续入局消息。

ownContext

SOAP 侦听器未传递上下文。返回的上下文反映用于运行侦听器的用户标识, 而不是用于创建原始请求消息的用户标识。

源上下文中的字段由队列管理器设置, 而不是由 SOAP 侦听器设置。

-n numThreads

`numThreads` 指定为 WebSphere MQ SOAP 侦听器生成的启动脚本中的线程数。缺省值为 10。如果消息吞吐量较高, 请考虑增大此数字。

-r

`-r` 指定替换任何现有请求或触发器监视器队列定义。仅当还指定了 `-tmq` 时, 才会替换触发器监视器队列。将使用标准缺省属性重新创建队列, 并且将删除队列上的现有消息。如果未使用 `-r` 选项, 那么不会更改任何现有队列定义, 也不会删除现有消息。通过不指定 `-r`, 可确保保留任何定制队列属性。

-s

配置侦听器以作为 WebSphere MQ 服务运行。如果同时指定了 `-s` 和 `-tmq`, 那么部署实用程序将显示错误消息并退出。

-tmp programName

`programName` 指定触发器监视器程序的名称。在 UNIX 或 Linux 环境中使用 `-tmp programName` 作为使用 `runmqtrm` 的替代方法。它启动的程序在 `mqm` 权限下运行。

例如:

```
amqwdeployMQService -f javaDemos/service/StockQuoteAxis.java
-tmq trigger.monitor.queue -tmp trigmon
```

-tmq queueName

`queueName` 指定触发器监视器队列名称。创建 IBM WebSphere MQ 进程定义以配置具有关联触发器监视器队列名称的 WebSphere MQ SOAP 侦听器的自动触发。如果未指定此选项, 那么部署实用程序不会定义任何触发配置。如果同时指定了 `-s` 和 `-tmq`, 那么部署实用程序将显示错误消息并退出。

URI 平台

请参阅 [第 885 页的『Web Service 部署的 URI 语法和参数』](#)。

-v

`-v` 设置来自外部命令的详细输出。始终显示错误消息。使用 `-v` 输出可定制以创建定制部署脚本的命令。

-w

`-w` 控制要生成的 WSDL 的样式。缺省值为 `rpcEnclosed`, 用于与先前发行版的 WebSphere MQ Transport for SOAP 兼容。使用 `rpcLiteral` 来创建与 Axis2 客户机代理生成兼容的 WSDL。`rpcEnclosed` 与 WS-I 建议不兼容。

-x 事务性

事务性 指定侦听器的事务控制类型。可以将事务性 设置为下列其中一个值:

onePhase

使用 IBM WebSphere MQ 一阶段支持。如果系统在处理期间失败, 那么会将请求消息重新传递到应用程序。WebSphere MQ 事务确保响应消息只写入一次。

twoPhase

使用两阶段支持。如果适当地编写服务, 那么将在服务的单个已落实执行中, 与其他资源协调, 仅传递一次消息。此选项仅适用于服务器绑定连接。

none

无事务支持。如果系统在处理期间失败，那么请求消息可能会丢失，即使是持久消息也是如此。该服务可能已执行，也可能未执行，并且可能已写入或未写入响应，报告或死信消息。

部署实用程序检查 `-x` 和 `-a` 标志的兼容性。请参阅 `-a` 标志的描述以获取详细信息。

错误

在 Windows 上，如果从 `amqswsd1` 报告了错误，请尝试发出以下命令以将 `.asmx` 文件注册为服务。

```
%windir%/Microsoft.NET/Framework/version number/aspnet_regiis.exe -ir
```

该问题通常发生在尚未安装 IIS 的系统上，或者 IIS 已安装在 NET 之后。`amqswsd1` 生成 `.wsdl` 文件时迁到该问题。

注：还需要注册表键以允许侦听器调用服务。如果您使用自己的定制部署过程，那么在运行时之前可能不会迁到问题。

来自 `amqdeployWMQService` 的输出文件

`amqdeployWMQService` 输出的目录和文件的列表

| 输出 | 描述 | 输出目录 | 文件名 |
|---|----------------------------|--|--|
| <code>.class</code> | 已编译的 Java 源文件 | <code>./generated/server/server package</code> | <code>classname.class</code> |
| <code>.wsdl</code> | 服务描述 (service description) | <code>./generated</code> | <code>classNameAxis_Wmq.wsdl</code> <code>classNameDotNet_Wmq.wsdl</code> |
| <code>.wsdd</code> | Axis 客户机和服务部署文件 | <code>./</code> | <code>client-config.wsdd</code> <code>server-config.wsdd</code> |
| | | <code>./generated/server/server package</code> | <code>className_deploy.wsdd</code> <code>className_undeploy.wsdd</code> |
| 客户机源 (<code>.vb</code> , <code>.cs</code> 和 <code>.java</code>) | .Net 客户机存根到 Axis 服务 | <code>./generated/client</code> | <code>classnameAxisService.cs</code> <code>classnameAxisService.vb</code> |
| | .Net 客户机存根到 .Net 服务 | <code>./generated/client</code> | <code>classnameDotNet.cs</code> <code>classnameDotNet.vb</code> |

表 583: 来自 **amqwdeployWMQService** 的输出文件 (继续)

| 输出 | 描述 | 输出目录 | 文件名 |
|--------------------------|---------------------|--|---|
| 客户机帮助程序 (.java 和 .class) | Java 客户机代理至 .Net 服务 | ./generated/server/soap/client/ remote/dotnetService | classNameDotNet.class classNameDotNet.java classNameDotNetLocator.class classNameDotNetLocator.java classNameDotNetSoap12Stub.class classNameDotNetSoap12Stub.java classNameDotNetSoap_BindingStub.class classNameDotNetSoap_BindingStub.java classNameDotNetSoap_PortType.class classNameDotNetSoap_PortType.java |
| | Java 客户机代理到 Axis 服务 | ./generated/server/soap/client/ remote/client package | SoapServerclassNameAxisBindingSoapStub.class SoapServerclassNameAxisBindingSoapStub.java classNameAxis.class classNameAxis.java classNameAxisService.class classNameAxisService.java classNameAxisServiceLocator.class classNameAxisServiceLocator.java |
| 脚本 (.cmd 和 .sh) | 侦听器脚本 | /generated/server | startWMQJListener.cmd startWMQJListener.sh startWMQNListener.cmd endWMQJListener.cmd endWMQJListener.sh endWMQNListener.cmd |

amqwdeployWMQService 的使用说明

描述 **amqwdeployWMQService** 执行的任务。

部署实用程序执行以下操作。

1. 检查以下文件的路径:
 - axis.jar.
 - WMQSOAP_HOME/java/lib/com.ibm.mq.soap.jar.
 - 在 Windows 上, csc.exe
2. 在 Windows 上, 使用 %SystemRoot%\Microsoft.NET\Framework\v1.1.432 或 (如果安装了 C# 编译器) csc.exe 的路径作为 .NET Framework 的路径。

注: 如果已安装 Microsoft Visual Studio 2008 (V 9), 那么 wsdl.exe 不在 csc.exe 的路径中。您需要将 .NET 框架的路径添加到路径变量; 例如:

```
Set Path=C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;%Path%
```

3. 创建 ./generated 目录和必需的子目录 (如果它们不存在)。
4. 对于 Java 服务, 将源编译为 className.class。
5. 生成 WSDL。
6. 对于 Java 服务, 创建部署描述符文件 className_deploy.wsdd 和 className_undeploy.wsdd
7. 对于 Java 服务, 创建或更新 Axis 部署描述符文件 server-config.wsdd。

8. 根据 WSDL 生成 Java, C# 和 Visual Basic 的客户机代理。

注: 在 Windows 上, 部署实用程序将生成 Visual Basic 和 C# 的代理, 而不考虑编写服务所用的语言。WSDL 及其生成的代理包含用于调用服务的相应 URI:

```
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
```

图 13: 生成的 .NET 客户机中用于调用 .NET 服务的示例 URI

```
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
```

图 14: 生成的 .NET 客户机中用于调用 Axis 1 服务的示例 URI

9. 编译 Java 代理。

10. 创建 WebSphere MQ 队列 *requestQueue* 以保存服务请求。缺省队列名称的格式为 *SOAPJ.directory*, 或者您可以在 *-u URI* 选项中指定 *requestQueue*。

11. 创建命令和 shell 脚本文件以启动用于处理请求队列的 WebSphere MQ SOAP 侦听器。

12. 如果已使用 *-tmq* 选项, 那么部署实用程序将创建 WebSphere MQ 定义以自动触发 WebSphere MQ SOAP 侦听器进程。

- 部署实用程序使用 **runmqsc** DEFINE PROCESS 命令的 APPLICID 属性来包含用于启动侦听器的命令。该命令中嵌入了部署目录的名称。APPLICID 字段的最大长度为 256, 这将限制部署目录的最大长度。Java 服务的目录限制如下:

- UNIX and Linux 系统: 218
- Windows: 197 减去请求队列名称的长度。

对于 .NET 服务, 目录限制如下所示:

- Windows: 209 减去服务名称的长度, 再减去 .asmx 扩展名。
- 部署实用程序将检查是否超过了 APPLICID 的限制。如果超出限制, 那么实用程序不会尝试定义触发过程。它会显示一条错误消息, 并且部署过程会失败, 而不会执行任何部署步骤。

以下示例显示了部署实用程序生成的用于启动 WebSphere MQ SOAP 侦听器的配置和启动命令。

```
DEFINE PROCESS(requestQueue) APPLICID(applicIDStr) REPLACE
ALTER QLOCAL (requestQueue) TRIGTYPE(FIRST) TRIGGER
PROCESS(requestQueue) INITQ(initQueueName) TRIGMPRI(0)
```

图 15: WebSphere MQ 配置命令以触发 SOAP 侦听器。

```
applicIDStr = start "Java WMQSoapListener -requestQueue"
                 /min .\generated\server\startWMQJListener.cmd;
```

图 16: 在 Windows 上启动轴 SOAP 侦听器

```
applicIDStr = start "WMQAsmxListener -className\
                 /min .\generated\server\startWMQNListener.cmd;
```

图 17: 在 Windows 上启动 .NET SOAP 侦听器

```
applicIDStr = xterm -iconic -T \"Java WMQSoapListener_requestQueue\"  
-e ./generated/server/startWMQJListener.sh & #
```

图 18: 在 UNIX and Linux 系统上启动轴 SOAP 侦听器

amqwRegisterdotNet: 注册 IBM WebSphere MQ Transport for SOAP to .NET

将 IBM WebSphere MQ Transport for SOAP 注册到 .NET 上的全局组合件高速缓存。

用途

amqwRegisterdotNet 向 .NET Framework 1 或 2 注册 WebSphere MQ SOAP 发送方，SOAP 侦听器和 WSDL 处理器。

语法

►► amqwRegisterdotNet ◄◄

描述

amqwRegisterdotNet 在安装期间自动运行。如果您使用的 .NET Framework 是在 WebSphere MQ Transport for SOAP 之前安装的，那么您无需再次运行该 .NET Framework。您可以根据需要多次运行此命令。使用它来重新注册具有不同 .NET Framework 版本的 WebSphere MQ Transport for SOAP。

注: 在 Windows 2003 Server 上，还必须运行 **aspnet_regiis** 实用程序，即使未部署到 Internet Information Server (IIS) 也是如此。aspnet_regiis.exe 实用程序的位置可能因 Microsoft .NET Framework 的不同版本而异，但通常位于 %SystemRoot%/Microsoft.NET/Framework/version number/aspnet_regiis 中。如果安装了多个版本，请将 **aspnet_regiis** 用于您正在使用的 .NET Framework 版本。

Apache 软件许可证

Apache 许可证 2.0 版，2004 年 1 月，<http://www.apache.org/licenses/>

<http://www.apache.org/licenses/>

Apache 许可证
版本 2.0，2004 年 1 月
<http://www.apache.org/licenses/>

使用，复制和分发的条款和条件

1. 定义。

"许可" 是指使用，复制的条款和条件，以及本文档的 1 到 9 部分所定义的分发。

"许可方" 是指版权所有者或由其授权的实体。授予许可证的版权所有者。

"法律实体" 是指代理实体和所有控制，控制或共同控制的其他实体对该实体进行控制。为本定义的目的，"控制" 是指 (i) 直接或间接造成指导或管理这类实体，无论是通过合同或否则，或者 (ii) 拥有 50% (50%) 或更多的 (iii) 该实体的实益所有权。

"您" (或 "您的") 是指个人或法律实体

行使此许可证授予的许可权。

"来源" 表格是指作出修改的首选表格, 包括但不限于软件源代码, 文档和配置文件。

"物体" 形式应是指由机械产生的任何形式 "源" 表单的变换或转换, 包括但不限于编译的对象代码, 生成的文档, 以及转换为其他媒体类型。

"工作" 是指作者的工作, 不论是在 "来源" 或 "根据许可证提供的对象表单, 如作品中包含或附加的版权声明 (附录中提供了一个示例)。

"衍生作品" 是指任何作品, 无论是在源或对象中表单, 该表单基于 (或派生自) "工作", 编辑修订, 注释, 阐述或其他修改代表了作为一个整体的原创作品。为此目的的本许可, 衍生作品不得包括剩余作品可分离或仅链接 (或以名称绑定) 到以下接口: 其作品及其衍生作品。

"贡献" 是指任何作者的作品, 包括工作的原始版本以及任何修改或添加该作品或其衍生作品, 即故意提交给许可方, 由版权所有者将其包含在工作中或由个人或法律实体授权代表版权所有者。为本定义的目的, "提交" 指发送的任何形式的电子, 口头或书面通信给许可方或其代表, 包括但不限于关于电子邮件列表, 源代码控制系统的通信, 以及由或代表许可人, 目的是讨论和改进工作, 但排除明显标记或其他方式的通信由版权拥有人以书面指定为 "非供稿"。

"贡献者" 是指许可方和任何个人或法律实体代表其已收到许可方和后来并入工作。

2. 授予版权许可证。在符合条款和条件的前提下本许可, 每位贡献者特此授予您永久, 全球, 非独占, 不收费, 免版税, 不可撤销复制, 准备衍生作品的版权许可, 公开显示, 公开执行, 再许可和分发以 "源" 或 "对象" 形式进行工作和此类派生工作。
3. 授予专利许可。在符合条款和条件的前提下本许可, 每位贡献者特此授予您永久, 全球, 非独占, 不收费, 免版税, 不可撤销 (除本部分另有规定外) 专利许可制作, 已制作, 使用, 提供销售, 销售, 导入和以其他方式转移工作, 其中该许可仅适用于可许可的专利申请被这些贡献者所侵犯的权利单独或通过其添加项的组合添加项提交此类贡献的工作。如果您

对任何实体提起专利诉讼 (包括诉讼中的交叉索赔或反诉), 指控工作或包含在工作中的贡献构成直接或分担性专利侵权, 则任何专利许可根据此工作许可证授予您的权利将终止截至该诉讼提出之日。

4. 重新分配。您可以复制和分发在任何媒介上的作品或其衍生作品, 无论有无修改, 并以 "源" 或 "对象" 形式提供, 前提是您满足以下条件:
 - (a) 你必须给予该工作的任何其他收件人或演绎作品本许可证的副本; 以及
 - (b) 你必须安排任何经修改的档案携带显眼的告示说明您已更改文件; 以及
 - (c) 您必须以任何衍生作品的源形式保留您分发的所有版权, 专利, 商标和来自工作源表单的归属通知, 不包括与以下内容的任何部分无关的声明: 演绎作品; 以及
 - (d) 如果工作包括作为其一部分的 "通知" 文本文件分发, 那么您分发的任何派生作品都必须包含归属声明的可读副本
在此类通知文件中, 不包括不包含的通知与衍生作品的任何部分相关, 至少在一个以下位置: 在已分发的 NOTICE 文本文件中作为衍生作品的一部分; 在源格式或文档 (如果与 "派生作品" 一起提供); 或在 "派生作品" 生成的显示中, 如果通常出现此类第三方声明的位置。内容 NOTICE 文件仅供参考, 并且请勿修改许可证。您可以添加自己的归属您分发的衍生作品中的声明, 以及或作为工作通知文本的附录, 提供无法解释此类额外的归属通知作为修改许可证。

您可以将自己的版权声明添加到您的修改中, 可能提供额外或不同的许可条款和条件以用于, 复制或分发您的修改, 或对于任何此类衍生作品整体而言, 只要您使用, 复制和分发工作, 否则符合本许可证中规定的条件。

5. 提交会费。除非您明确声明, 否则有意提交以包含在工作中的任何贡献您向许可方提供的服务应符合以下条款和条件: 本许可, 不附带任何其他条款或条件。尽管有上述规定, 此处的任何内容都不得取代或修改您可能已执行的任何单独许可协议的条款与许可方就此类捐款进行了讨论。
6. 商标。此许可证不授予使用该行业的许可权许可方的名称, 商标, 服务标记或产品名称,

但为合理和习惯地用于描述工作的原点并复制 NOTICE 文件的内容。

7. 保修免责声明。除非适用法律要求或经书面同意，许可方提供工作 (以及每个工作贡献者在 "按现状" 的基础上提供其贡献)，不附有任何种类的保证或条件，无论是明示的或默示，包括但不限于任何保证或条件所有权，非侵权，适销性或适销性特定用途。您单独负责确定使用或重新分发工作的适当性，并承担任何与您在此许可证下执行许可权相关的风险。
8. 责任限制。在任何情况下，在任何法律理论下，无论是在侵权 (包括过失)，合同还是其他方面，除非适用法律 (如故意和严重过失行为) 或以书面同意，任何贡献者应对您承担损害赔偿赔偿责任，包括任何直接的，间接的，特殊的，任何性质的附带或后果性损害本许可的结果或出于使用或无法使用工作 (包括但不限于商誉损失损害赔偿，停工，计算机故障或故障，或任何和所有其他商业损害或损失)，即使此种捐助人已被告知这种损害的可能性。
9. 接受担保或其他责任。重新分发时作品或衍生作品，您可以选择提供，并收取费用，接受支持，保证，赔偿，或其他责任义务和/或权利符合本许可证。但是，在接受此类义务时，您只能采取行动以自己的名义和独自负责，而不是代表任何其他贡献者，并且只有在您同意赔偿的情况下，进行辩护，并让每个 "贡献者" 对任何责任都无害由该贡献者引起或针对该贡献者提出的索赔 (按原因) 您接受任何此类保证或其他责任。

条款及条件结束

附录: 如何将 Apache 许可证应用于您的工作。

要将 Apache 许可证应用于您的工作，请附加以下内容
样板通知，用括号 "[]" 括起来的字段
替换为您自己的识别信息。(不包含
支架!) 应在适当的文件中附上案文。
文件格式的注释语法。我们还建议
文件或类名以及用途描述将包含在
与版权声明相同的 "打印页面" 更容易
第三方归档中的标识。

版权 [yyyy] [版权所有者的名称]

根据 Apache License , Version 2.0 ("License") 获得许可;
除非符合许可证，否则不得使用此文件。
您可以在以下位置获取许可证副本:

<http://www.apache.org/licenses/LICENSE-2.0>

除非适用法律要求或书面同意，软件

根据 "按现状" 分发许可证,
不附有任何种类的明示或默示的保证或条件。
请参阅许可证以获取特定语言管理许可权和
许可证下的限制。

MQMD SOAP 设置

IBM WebSphere MQ SOAP 发送方和 IBM WebSphere MQ SOAP 侦听器创建消息描述符 (**MQMD**)。本主题描述了在创建自己的 SOAP 发送方或侦听器时必须在 MQMD 中设置的字段。

用途

MQMD 中设置的值控制 IBM WebSphere MQ SOAP 发送方, IBM WebSphere MQ SOAP 侦听器和 SOAP 客户机程序之间的消息交换。如果您创建自己的 SOAP 发送方或侦听器, 请遵循 [第 865 页的表 584](#) 中的规则。

描述

[第 865 页的表 584](#) 描述了 IBM WebSphere MQ SOAP 发送方和 IBM WebSphere MQ SOAP 侦听器如何设置 **MQMD** 字段。如果您编写自己的发送方或侦听器, 那么必须根据交换消息的规则来设置这些字段。IBM WebSphere MQ SOAP 侦听器符合典型的 IBM WebSphere MQ 消息交换协议。如果您编写自己的发送方以使用 IBM WebSphere MQ SOAP 侦听器, 那么可以设置不同的 **MQMD** 值。

在 [第 865 页的表 584](#) 中, 设置列中的值按如下所示进行组织:

请求, 单向

由 IBM WebSphere MQ SOAP 发送方进行的设置。

响应, 报告

IBM WebSphere MQ SOAP 侦听器为响应 IBM WebSphere MQ SOAP 发送方请求而进行的设置。

ALL

由 IBM WebSphere MQ SOAP 发送方和 IBM WebSphere MQ SOAP 侦听器进行的设置。

定制发送方

您可以编写自己的发件人。通常, 定制发送方会覆盖标准报告选项。

| 字段名称 | 设置 | 值 |
|----------------|---|----------|
| <i>StrucId</i> | ALL MQMD_STRUC_ID | 'MD' 1 |
| <i>Version</i> | ALL MQMD_VERSION_2 | 2 |
| <i>Report</i> | ALL MQRO_NONE + MQRO_NEW_MSG_ID + MQRO_COPY_MSG_ID_TO_CORREL_ID + MQRO_EXCEPTION + MQRO_EXPIRY + MQRO_DISCARD 定制发送方 请参阅 第 869 页的『定制报告选项』 | 52428800 |

表 584: MQMD SOAP 设置 (继续)

| 字段名称 | 设置 | 值 |
|-----------------------|--|--|
| <i>MsgType</i> | 请求 MQMT_REQUEST 响应 MQMT_REPLY 报告 MQMT_REPORT 单向 MQMT_DATAGRAM | MQMT_REQUEST 1 MQMT_REPLY 2 MQMT_REPORT 4 MQMT_DATAGRAM 8 |
| <i>Expiry</i> | 请求, 单向 由 URI 中的 到期 选项指定。缺省值为 MQEI_UNLIMITED。 响应 请求消息中 到期 的值 报告 MQEI_UNLIMITED | MQEI_UNLIMITED -1 |
| <i>Feedback</i> | 请求, 响应, 单向 MQFB_NONE. 报告 <ul style="list-style-type: none"> 由根据正常规则设置的队列管理器值生成。 由 IBM WebSphere MQ SOAP 侦听器生成: MQRC_BACKOUT_THRESHOLD_REACHED 已超过多次尝试的回退阈值。 MQRCCF_MD_FORMAT_ERROR 消息未识别为具有 MQRFH2 头。 MQRC_RFH_PARM_MISSING MQRFH2 中缺少必需参数, 例如 SoapAction。 MQRC_RFH_FORMAT_ERROR MQRFH2 的基本完整性检查失败, 例如, 内部长度已损坏。 MQRC_RFH_ERROR MQRFH2 通过了完整性检查, 但消息主体未设置为 MQFMT_NONE。 | MQFB_NONE 0 MQRC_BACKOUT_THRESHOLD_REACHED 2362 MQRCCF_MD_FORMAT_ERROR 3023 MQRC_RFH_PARM_MISSING 2339 MQRC_RFH_FORMAT_ERROR 2421 MQRC_RFH_ERROR 2334 |
| <i>Encoding</i> | ALL MQENC_NATIVE | 取决于环境 |
| <i>CodedCharSetId</i> | ALL 设置为 UTF-8 | 1208 |

表 584: MQMD SOAP 设置 (继续)

| 字段名称 | 设置 | 值 |
|---------------------|---|--|
| <i>Format</i> | <p>请求, 响应, 单向 MQFMT_RF_HEADER_2</p> <p>报告 队列管理器报告 遵循 IBM WebSphere MQ 规则</p> <p>IBM WebSphere MQ SOAP 侦听器报告 原始请求消息的格式。</p> | <p>MQFMT_RF_HEADER_2 "MQRFH2 "</p> |
| <i>Priority</i> | <p>请求, 单向 由 URI 中的 优先级 选项指定。缺省值为 MQPRI_PRIORITY_AS_Q_DEF。</p> <p>响应, 报告 请求消息中 优先级 的值。</p> | <p>MQPRI_PRIORITY_AS_Q_DEF -1</p> |
| <i>Persistence</i> | <p>请求, 单向 MQPER_PERSISTENCE_AS_Q_DEF.</p> <p>响应, 报告 请求消息中 持久性 的值。</p> | <p>MQPER_PERSISTENCE_AS_Q_DEF 2</p> |
| <i>MsgId</i> | <p>请求, 单向 由队列管理器生成。</p> <p>响应, 报告 将生成 IBM WebSphere MQ SOAP 发送方集 MQRO_NEW_MSG_ID 和 <i>MsgId</i></p> | <p>已生成 队列管理器生成的唯一值</p> |
| <i>CorrelId</i> | <p>请求, 单向, 报告 MQCI_NONE</p> <p>响应, 报告 IBM WebSphere MQ SOAP 发送方集 MQRO_COPY_MSG_ID_TO_CORREL_ID 和侦听器从请求消息复制 <i>MsgId</i>。</p> | <p>MQCI_NONE 0</p> |
| <i>BackoutCount</i> | <p>ALL 未使用</p> | <p>0</p> |
| <i>ReplyToQ</i> | <p>请求 由 URI 中的 replyDestination 选项指定。缺省值为 SYSTEM.SOAP.RESPONSE.QUEUE。</p> <p>响应, 单向, 报告 留空</p> | |
| <i>ReplyToQMGr</i> | <p>ALL 字段留空</p> | <p>由队列管理器生成; 请参阅 应答队列和队列管理器。</p> |

表 584: MQMD SOAP 设置 (继续)

| 字段名称 | 设置 | 值 |
|-------------------------|--|---|
| <i>UserIdentifier</i> | <p>请求, 单向, 报告 留空</p> <p>响应 取决于提供给侦听器的 <code>-i passContext</code> 选项以及侦听器运行所使用的权限。</p> | <p>请求, 单向, 报告 由队列管理器生成; 请参阅第 401 页的『UserIdentifier (MQCHAR12)』。</p> <p>响应 <i>variable</i></p> |
| <i>AccountingToken</i> | <p>ALL MQACT_NONE</p> | <p>MQACT_NONE 空字符串或空白</p> <p>由队列管理器设置; 请参阅第 367 页的『AccountingToken (MQBYTE32)』。</p> |
| <i>ApplIdentityData</i> | <p>ALL None</p> | 空字符串或空白 ² |
| <i>PutApplType</i> | <p>ALL MQAT_NO_CONTEXT</p> | <p>MQAT_NO_CONTEXT 0</p> <p>队列管理器生成的值; 请参阅第 390 页的『PutApplType (MQLONG)』。</p> |
| <i>PutApplName</i> | <p>ALL None</p> | 队列管理器生成的值; 请参阅第 389 页的『 PutAppl 名称 (MQCHAR28) 』。 |
| <i>PutDate</i> | <p>ALL None</p> | 队列管理器生成的值; 请参阅第 391 页的『 PutDate (MQCHAR8) 』。 |
| <i>PutTime</i> | <p>ALL None</p> | 队列管理器生成的值; 请参阅第 392 页的『 PutTime (MQCHAR8) 』。 |
| <i>ApplOriginData</i> | <p>ALL None</p> | 空字符串或空白 ² |
| <i>GroupId</i> | <p>请求, 单向, 报告 MQGI_NONE</p> <p>响应 从请求消息复制字段</p> | Null |
| <i>MsgSeqNumber</i> | <p>请求, 单向, 报告 未使用</p> <p>响应 从请求消息复制字段</p> | 由队列管理器生成; 请参阅 队列上的物理顺序 。 |
| <i>Offset</i> | <p>请求, 单向, 报告 未使用</p> <p>响应 从请求消息复制字段</p> | 0 |

表 584: MQMD SOAP 设置 (继续)

| 字段名称 | 设置 | 值 |
|--|---|---|
| <i>MsgFlags</i> | 请求, 单向, 报告 MQMF_NONE 响应 从请求消息复制字段 | MQMF_NONE 0 请参阅第 381 页的『 MsgFlags (MQLONG) 』 |
| <i>OriginalLength</i> | 请求, 单向, 响应 MQOL_UNDEFINED 报告 原始请求消息的长度 | MQOL_UNDEFINED -1 |
| 注意: 1. 符号 \ 表示单个空白字符。 2. 值 空字符串或空白 表示 C 中的空字符串, 其他编程语言中的空白字符。 | | |

定制报告选项

您可以编写自己的 SOAP 发送方, 并将其与提供的侦听器配合使用。通常, 您可以编写发件人以更改报告选项的选择。IBM WebSphere MQ SOAP 侦听器支持报告选项的大多数组合, 如以下列表中所述。

- IBM WebSphere MQ SOAP 侦听器支持的报告选项:
 - 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
- 队列管理器支持的报告选项:
 - 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
- IBM WebSphere MQ SOAP 侦听器不支持以下报告选项。
 - MQRO_PAN
 - MQRO_NAN

用途

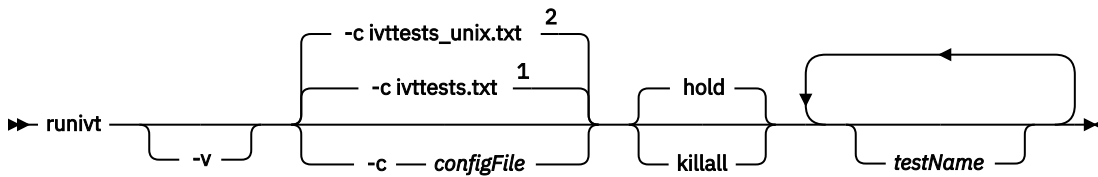
runivt 命令使用随 WebSphere MQ Transport for SOAP 提供的样本程序将 Web Service 请求从客户机发送到服务。它运行 Axis 1.4, .NET Framework 1 和 .NET Framework 2 的测试。在测试脚本文件中配置测试。Windows 的缺省测试脚本文件在 Java 和 .NET 客户机与服务之间运行测试组合。

描述

runivt 必须从其自己的目录运行。

该命令在另一个命令窗口中启动侦听器。因此，必须从 UNIX and Linux 系统上的 X Window System 会话运行该命令。

runivt syntax



注:

¹ Default on Windows

² Default on UNIX and Linux systems

runivt 参数

-v

详细方式。将更完整的错误消息写入控制台。

-c configFile

用于定义要运行的测试的配置文件。缺省情况下，将使用 Windows, UNIX 或 Linux 系统随附的缺省配置文件。

hold

在测试完成后保持侦听器处于运行状态

killall

测试完成时结束侦听器

testName

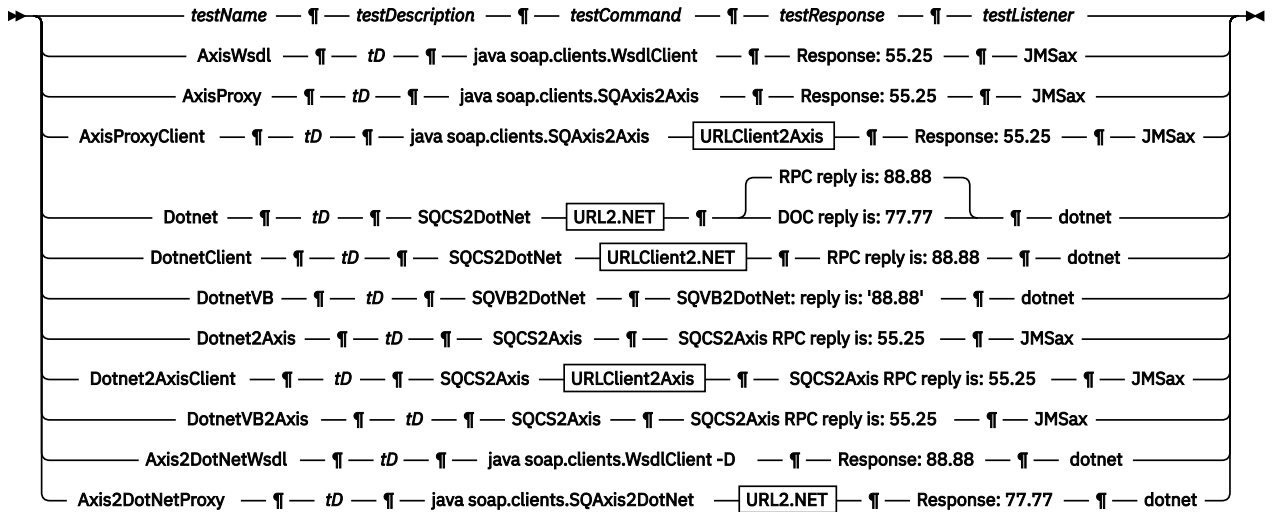
要运行的测试的空格分隔列表。从配置文件中选择测试名称。如果未指定名称，那么将运行配置文件中的所有测试。

Configuration file

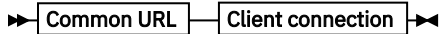
Each configuration file parameter is a separate line of the file. Leave a blank line between each group of parameters.

The parameters in the `ivttests.txt` parameter file are listed.

configFile syntax



URLClient2Axis



URL2.NET



URLClient2.NET



Common URL

► jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM — & — initialContextFactory — = — ►

► com.ibm.mq.jms.Nojndi — & — connectionFactory — = — ►

► connectQueueManager — (— WMQSOAP.DEMO.QM —) — ►

Client connection

► clientConnection — (— localhost%25289414WMQSOAP.DEMO.QM%2529 —) — clientChannel — ►

► (— TESTCHANNEL —) — ►

Target service

► & — targetService — = — StockQuoteDotNet.asmx — ►

configFile 参数

testName

测试的名称。在 **runivt** 命令中使用 *testName*

testDescription

有关测试的 Documentation

testCommand

runivt 命令执行的用于发出客户机请求的命令。

testResponse

客户机请求返回到控制台的准确响应字符串。要使测试成功，*testResponse* 必须与实际响应匹配。

testListener

由 **runivt** 启动以处理 SOAP 请求的 WebSphere MQ SOAP 侦听器的名称。dotnet 和 JMSax 是提供的侦听器 **amqwsOAPNETlistener** 和 **SimpleJavaListener** 的同义词。

示例

```
runivt
```

图 19: 运行所有缺省测试

```
runivt dotnet
```

图 20: 从缺省测试运行特定测试

```
runivt -c mytests.txt
```

图 21: 运行一组定制测试

相关信息

[验证 WebSphere MQ Transport for SOAP](#)

通过 IBM WebSphere MQ Transport for SOAP 保护 Web Service

您可以通过两种方法之一来保护使用 IBM WebSphere MQ Transport for SOAP 的 Web Service。在客户机与服务器之间创建 SSL 通道，或者使用 Web Service 安全性。

SSL 和 WebSphere MQ Transport for SOAP

WebSphere MQ Transport for SOAP 提供了多个 SSL 选项，可以指定这些选项用于配置为以 SSL 方式运行的客户机通道。这些选项在 .NET 和 Java 环境之间有所不同。WebSphere MQ SOAP 发送方和侦听器仅处理适用于其特定环境的 SSL 选项。它们会忽略不适用的选项。

是否存在针对 .NET 客户机的 `sslCipherSpec` 选项以及针对 Java 客户机的 `sslCipherSuite` 选项可确定是否使用 SSL。如果未在 URI 中指定该选项，那么缺省情况下不会使用 SSL，并且将忽略所有其他 SSL 选项。除指示的选项外，所有 SSL 选项都是可选的。

对于 WebSphere MQ 客户机，请在 URI 或通道定义表中设置 SSL 属性。在服务器上，使用 WebSphere MQ 的工具设置属性。

缺省情况下，在通道上启用 SSL 时，将设置标准 WebSphere MQ SSL 选项 `SSLCAUTH`。客户机必须先认证自身，然后才能开始 SSL 通信。如果未设置 `SSLCAUTH`，那么将在不进行客户机认证的情况下建立 SSL 通信。

要认证自己，客户机必须在其密钥存储库中分配队列管理器可接受的证书。为了实现其他安全性，可以将 WebSphere MQ 通道配置为仅接受受限列表中的证书。通过根据通道的对等名称属性检查证书的专有名称来限制列表。

如果使用 Java，那么来自 WebSphere MQ SOAP 客户机的第一个 SSL 连接将导致修复以下 SSL 参数。在使用同一客户机进程的后续连接中使用相同的值：

- `sslKey` 存储
- `sslKeyStorePassword`
- `sslTrust` 存储
- `sslTrustStorePassword`
- `sslFipsRequired`
- `sslLDAPCRLservers`

未定义更改这些参数对来自此客户机的后续连接的影响。

如果使用 .NET，那么来自 WebSphere MQ SOAP 客户机的第一个 SSL 连接将导致修正以下 SSL 参数。在使用同一客户机进程的后续连接中使用相同的值：

- `SSLKeyRepository`
- `SSLCryptoHardware`

- sslFipsRequired
- sslLDAPCRLservers

未定义更改这些参数对来自此客户机的后续连接的影响。如果所有 SSL 连接都变为不活动状态并且建立了新的 SSL 连接，那么将重置这些参数。

还可以将以下属性指定为系统属性:

- sslKey 存储
- sslKeyStorePassword
- sslTrust 存储
- sslTrustStorePassword

如果将它们同时指定为系统属性和 URI 中的值，并且值不同，那么部署实用程序将显示警告。URI 值优先。

相关任务

指定运行时在 MQI 客户机上仅使用经过 FIPS 认证的 CipherSpecs

相关参考

WebSphere MQ Web Service URI 中的 SSL 连接工厂参数

将 SSL 选项添加到 IBM WebSphere MQ Web Service URI 中的连接工厂选项列表。

适用于 UNIX, Linux 和 Windows 的联邦信息处理标准 (FIPS)

WebSphere MQ Web Service URI 中的 SSL 连接工厂参数

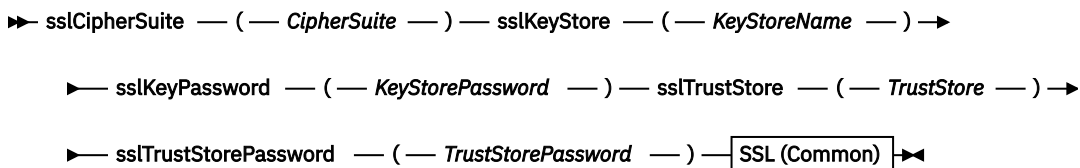
将 SSL 选项添加到 IBM WebSphere MQ Web Service URI 中的连接工厂选项列表。

用途

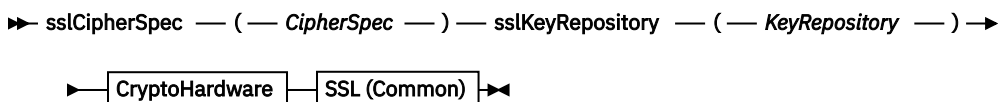
可以在 IBM WebSphere MQ Web Service 客户机与托管 Web Service 的队列管理器之间使用安全连接。SSL 选项控制如何在 IBM WebSphere MQ MQI 客户机/服务器通道连接上配置 SSL。

Syntax diagram

SSL (Java)

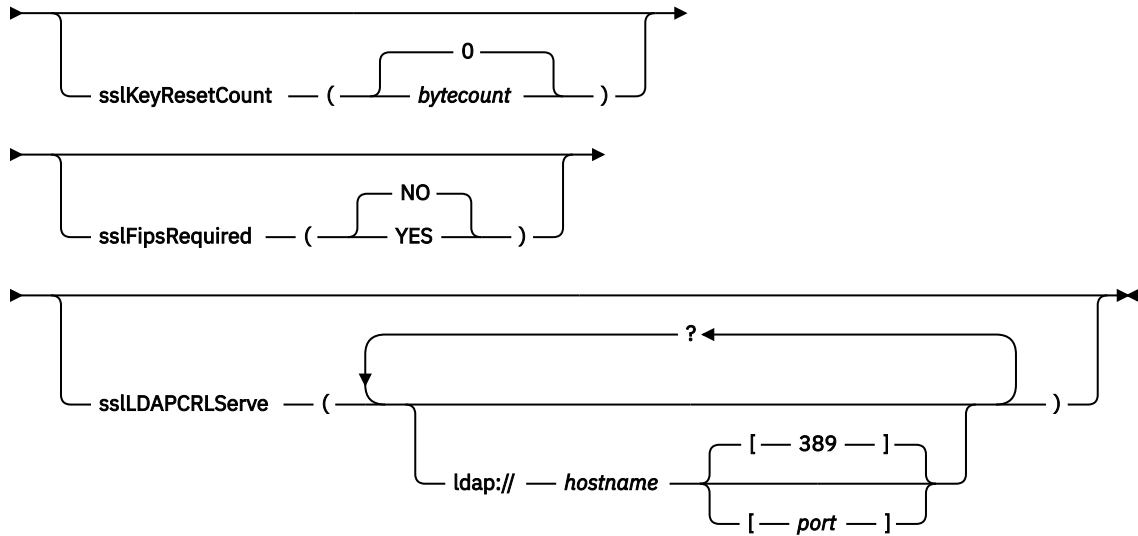


SSL (.NET)



SSL (Common)

►► sslCipherPeerName — (— PeerName —) →



CryptoHardware

►► sslCryptoHardware — = — PKCS #11 Path and file name — ; — PKCS #11 token label — ; →

► — PKCS #11 token password — ; — symmetric cipher setting — ; ►

必需的 SSL 参数 (公共)

sslPeerName (*peerName*)

peerName 指定通道上使用的 sslPeer 名称。

必需的 SSL 参数 (Java)

sslCipherSuite (*CipherSuite*)

CipherSuite 指定通道上使用的 sslCipherSuite。客户机指定的 CipherSuite 必须与服务器连接通道上指定的 CipherSuite 相匹配。

sslKeyStore (*KeyStore* 名称)

KeyStore 名称 指定通道上使用的 sslKeyStoreName。密钥库包含用于向服务器认证客户机的客户机专用密钥。如果 SSL 连接配置为接受匿名客户机连接，那么密钥库是可选的。

sslKeyStorePassword (*KeyStore* 密码)

KeyStore 密码 指定通道上使用的 sslKeyStorePassword。

sslTrustStore (*TrustStore* 名称)

TrustStore 名称 指定通道上使用的 sslTrustStoreName。信任库保存服务器的公用证书或其密钥链，以向客户机认证服务器。如果使用认证中心的根证书来认证服务器，那么信任库是可选的。在 Java 中，根证书保存在 JRE 证书库 cacerts 中。

sslTrustStorePassword (*TrustStore* 密码)

TrustStorePassword 指定通道上使用的 sslTrustStorePassword。

必需的 SSL 参数 (.NET)

sslCipherSpec (*CipherSpec*)

CipherSpec 指定通道上使用的 sslCipherSpec。如果指定了此选项，那么将在客户机通道上使用 SSL。

sslKeyRepository (*KeyRepository*)

KeyRepository 指定存储 SSL 密钥和证书的通道上使用的 sslCipherSpec。*KeyRepository* 以主干格式指定，即，具有文件名但省略了文件扩展名的完整路径。设置 sslKeyRepository 的效果与在 MQCONN 调用上设置 MQSCO 结构中的 KeyRepository 字段相同。

可选 SSL 参数 (.NET)

sslCryptoHardware (CryptoHardware)

CryptoHardware 指定通道上使用的 sslCryptoHardware。此字段的可能值及其设置效果与 MQCONNX 上 MQSCO 结构的 CryptoHardware 字段相同。

可选 SSL 参数 (公共)

sslKeyResetCount (bytecount)

bytecount 指定在必须重新协商 SSL 密钥之前通过 SSL 通道传递的字节数。要禁用 SSL 密钥的重新协商，请省略该字段或将其设置为零。零是某些环境中唯一受支持的值，请参阅 [重新协商 WebSphere MQ classes for Java](#) 中的密钥。设置 sslKeyResetCount 的效果与设置 MQCONNX 调用的 MQSCO 结构中的 KeyResetCount 字段相同。

sslFipsRequired (fipsCertified)

fipsCertified 指定 *CipherSpec* 还是 *CipherSuite* 必须在通道上的 IBM WebSphere MQ 中使用 FIPS 认证的密码术。设置 *fipsCertified* 的效果与在 MQCONNX 调用上设置 MQSCO 结构的 FipsRequired 字段相同。

sslLDAPCRLServers (LDAPServerList)

LDAPServerList 指定要用于证书撤销列表检查的 LDAP 服务器的列表。

对于启用 SSL 的客户机连接，*LDAPServerList* 是要用于证书撤销列表 (CRL) 检查的 LDAP 服务器的列表。将针对列出的某个 LDAP CRL 服务器检查队列管理器提供的证书；如果找到该证书，那么连接将失败。将依次尝试每个 LDAP 服务器，直到与其中一个服务器建立连接为止。如果无法连接到任何服务器，那么将拒绝该证书。成功建立与其中一个服务器的连接后，将根据该 LDAP 服务器上存在的 CRL 接受或拒绝该证书。

如果 *LDAPServerList* 为空，那么不会针对证书撤销列表检查属于队列管理器的证书。如果提供的 LDAP URI 列表无效，那么将显示错误消息。设置此字段的的效果与包含 MQAIR 记录并从 MQCONNX 上的 MQSCO 结构访问这些记录的效果相同。

相关任务

[指定运行时在 MQI 客户机上仅使用经过 FIPS 认证的 CipherSpecs](#)

相关参考

[SSL 和 WebSphere MQ Transport for SOAP](#)

WebSphere MQ Transport for SOAP 提供了多个 SSL 选项，可以指定这些选项用于配置为以 SSL 方式运行的客户机通道。这些选项在 .NET 和 Java 环境之间有所不同。WebSphere MQ SOAP 发送方和侦听器仅处理适用于其特定环境的 SSL 选项。它们会忽略不适用的选项。

[适用于 UNIX，Linux 和 Windows 的联邦信息处理标准 \(FIPS\)](#)

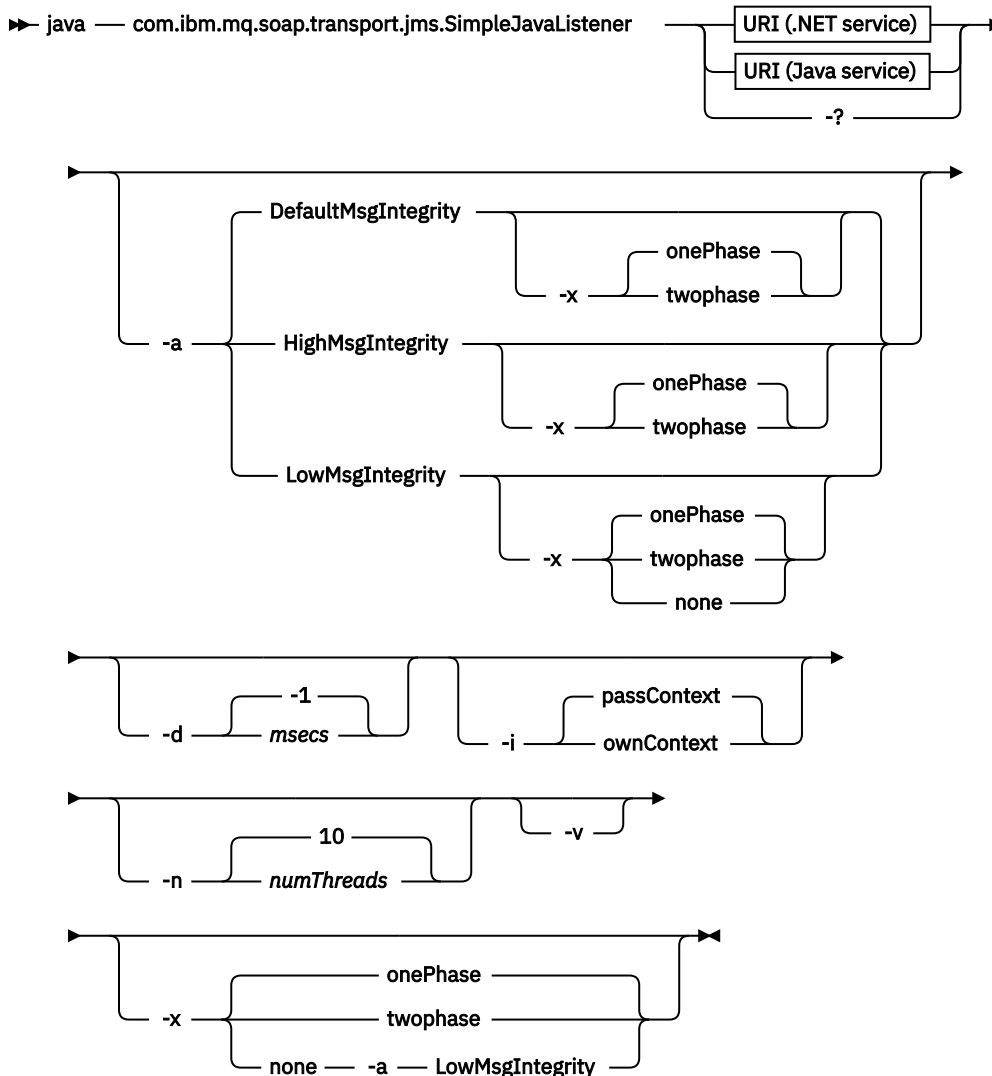
SimpleJavaListener: IBM WebSphere MQ Axis 的 SOAP 侦听器 1.4

IBM WebSphere MQ SOAP Axis 侦听器 1.4 的语法和参数。

用途

启动 IBM WebSphere MQ Axis 的 SOAP 侦听器 1.4。

Java



必需参数

URI 平台

请参阅第 885 页的『Web Service 部署的 URI 语法和参数』。

-?

打印描述如何使用命令的帮助文本。

可选参数

-a integrityOption

integrityOption 指定 WebSphere MQ SOAP 侦听器的行为 (如果无法将失败请求消息放入死信队列)。
integrityOption 可以采用下列其中一个值:

DefaultMsgIntegrity

对于非持久消息, 侦听器将显示一条警告消息, 并在废弃原始消息的情况下继续执行。对于持久消息, 它会显示错误消息, 回退请求消息以使其保留在请求队列上并退出。如果省略了 -a 选项, 或者未指定 *integrityOption*, 那么 DefaultMsgIntegrity 适用。

LowMsgIntegrity

对于持久消息和非持久消息, 侦听器会显示警告并继续执行, 从而废弃该消息。

HighMsgIntegrity

对于持久消息和非持久消息, 侦听器会显示一条错误消息, 回退请求消息以使其保留在请求队列中并退出。

部署实用程序检查 **-x** 和 **-a** 标志的兼容性。如果指定了 **-x none**，那么必须指定 **-a LowMsgIntegrity**。如果标志不兼容，那么部署实用程序将退出并返回错误消息，并且未执行任何部署步骤。

-d 毫秒

msecs 指定在任何线程上接收到请求消息时，WebSphere MQ SOAP 侦听器保持活动的毫秒数。如果 *msecs* 设置为 **-1**，那么侦听器将无限期保持活动状态。

-i 上下文

上下文 指定侦听器是否传递身份上下文。上下文 采用以下值：

passContext

将原始请求消息的身份上下文设置到响应消息中。SOAP 侦听器检查其是否有权从请求队列保存上下文并将其传递到响应队列。当打开请求队列以保存上下文和响应队列以传递上下文时，它会在运行时进行检查。如果它没有必需的权限，或者 **MQOPEN** 调用失败，并且未处理响应消息。响应消息将与包含来自失败 **MQOPEN** 的返回码的死信头一起放在死信队列上。然后，侦听器继续正常处理后续入局消息。

ownContext

SOAP 侦听器未传递上下文。返回的上下文反映用于运行侦听器的用户标识，而不是用于创建原始请求消息的用户标识。

源上下文中的字段由队列管理器设置，而不是由 SOAP 侦听器设置。

-n numThreads

numThreads 指定为 WebSphere MQ SOAP 侦听器生成的启动脚本中的线程数。缺省值为 **10**。如果消息吞吐量较高，请考虑增大此数字。

-v

-v 设置来自外部命令的详细输出。始终显示错误消息。使用 **-v** 输出可定制以创建定制部署脚本的命令。

-w serviceDirectory

serviceDirectory 是包含 Web Service 的目录。

-x 事务性

事务性 指定侦听器的事务控制类型。可以将 事务性 设置为下列其中一个值：

onePhase

使用 IBM WebSphere MQ 一阶段支持。如果系统在处理期间失败，那么会将请求消息重新传递到应用程序。WebSphere MQ 事务确保响应消息只写入一次。

twoPhase

使用两阶段支持。如果适当地编写服务，那么将在服务的单个已落实执行中，与其他资源协调，仅传递一次消息。此选项仅适用于服务器绑定连接。

none

无事务支持。如果系统在处理期间失败，那么请求消息可能会丢失，即使是持久消息也是如此。该服务可能已执行，也可能未执行，并且可能已写入或未写入响应，报告或死信消息。

部署实用程序检查 **-x** 和 **-a** 标志的兼容性。请参阅 **-a** 标志的描述以获取详细信息。

Java 示例

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi"
-n 20
```

WebSphere MQ SOAP 侦听器

WebSphere MQ SOAP 侦听器从指定为 URI 中的目标的队列中读取入局 SOAP 请求。它检查请求消息的格式，然后使用 Web Service 基础结构来调用 Web Service。WebSphere MQ SOAP 侦听器使用 URI 中的应答目标队列从 Web Service 返回任何响应或错误。它将 WebSphere MQ 报告返回到应答队列。

此处使用标准 Web Service 意义上的术语“侦听器”。它与 **runmq1sr** 命令调用的标准 WebSphere MQ 侦听器不同。

描述

Java SOAP 侦听器作为 Java 类实现，并使用 Axis 1.4 运行服务。 .NET 侦听器是控制台应用程序，并运行 .NET Framework 1 或 .NET Framework 2 服务。对于 .NET Framework 3 服务，请使用适用于 Microsoft Windows Communication Foundation (WCF) 的 WebSphere MQ 定制通道。

部署实用程序创建脚本以自动启动 Java 或 .NET SOAP 侦听器。可以使用 **amqSOAPNETListener** 命令或通过调用 **SimpleJavaListener** 类来手动启动 SOAP 侦听器。您可以通过在部署实用程序中设置 **-s** 选项来配置要作为 WebSphere MQ 服务启动的 WebSphere MQ SOAP 侦听器。或者，使用触发来启动侦听器，或者使用部署实用程序生成的启动和结束侦听器脚本。您可以手动配置触发，也可以使用 **-tmq** 和 **-tmp** 部署选项来自动配置触发。您可以通过将请求队列设置为 **GET (DISABLED)** 来结束侦听器。

| Web Service 基础结构 | UNIX and Linux 系统 | Windows Java | Windows .NET |
|------------------|------------------------------|-------------------------------|-------------------------------|
| 启动侦听器 | startWMQJListener.sh | startWMQJListener.cmd | startWMQNListener.cmd |
| 停止侦听器 | endWMQJListener.sh | endWMQJListener.cmd | endWMQNListener.cmd |
| 定义侦听器服务 | defineWMQJListener.sh | defineWMQJListener.cmd | defineWMQNListener.cmd |

WebSphere MQ SOAP 侦听器将 **endpointURL** 和 **soapAction** 字段从 SOAP 消息传递到 SOAP 基础结构。侦听器通过 Web Service 基础结构调用服务并等待响应。侦听器不会验证 **endpointURL** 和 **soapAction**。这些字段由 WebSphere MQ SOAP 发送方根据 SOAP 客户机在 URI 集中提供的数据进行设置。

侦听器创建响应消息并将其发送到请求消息 URI 中提供的应答目标。此外，侦听器根据请求消息中的报告选项在响应消息中设置相关标识。它从请求消息返回到期，持久性和优先级设置。在某些情况下，侦听器还会将报告消息发送回客户机。

如果 SOAP 请求中存在格式错误，那么侦听器将使用应答目标队列向客户机返回报告消息。如果已请求报告，那么队列管理器还会使用应答目标队列向客户机返回报告消息。将完整的报告消息写入响应队列，以响应许多事件：

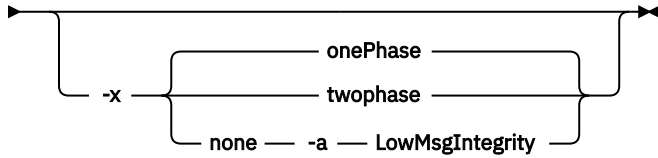
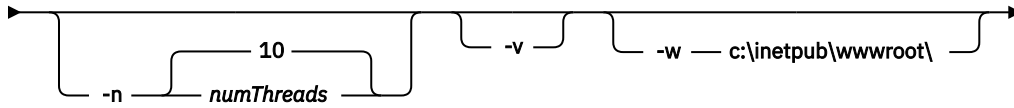
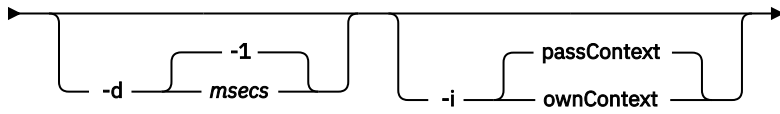
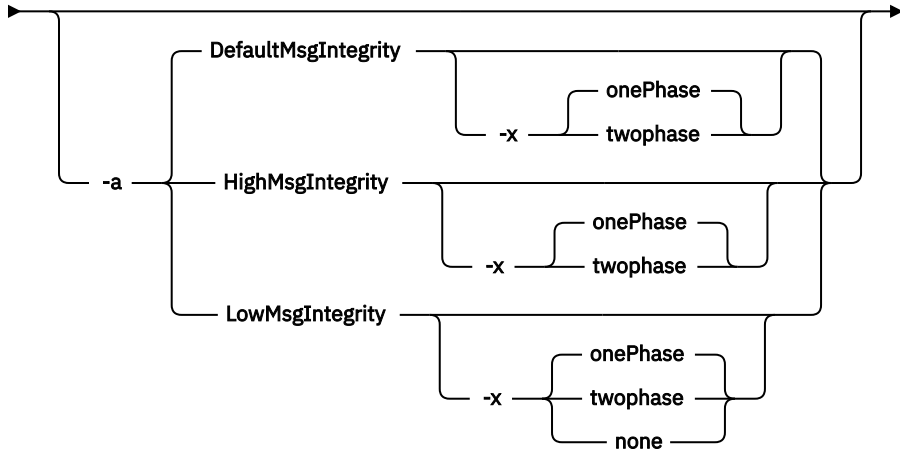
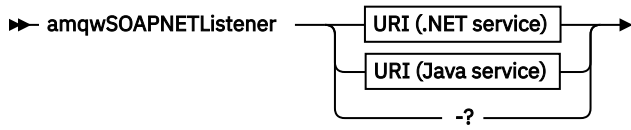
- 异常。
- 消息到期。
- 无法识别请求消息的格式。
- **MQRFH2** 头的完整性检查失败。
- 主消息体的格式不是 **MQFMT_NONE**。
- 当 WebSphere MQ SOAP 侦听器正在处理请求时，超出回退/重试阈值。

WebSphere MQ SOAP 发送方集 **MQRO_EXCEPTION_WITH_FULL_DATA** 和 **MQRO_EXPIRATION_WITH_FULL_DATA** 报告选项。由于 WebSphere MQ SOAP 发送方设置了报告选项，因此报告消息包含整个原始请求消息。WebSphere MQ SOAP 发送方还会设置 **MQRO_DISCARD** 选项，这将导致在返回报告消息后废弃该消息。如果报告选项不满足您的需求，请编写您自己的发件人以使用不同的 **MQRO_EXCEPTION** 和 **MQRO_DISCARD** 报告选项。如果 SOAP 请求由未设置 **MQRO_DISCARD** 的其他发送方发送，那么会将失败消息写入死信队列 (DLQ)。

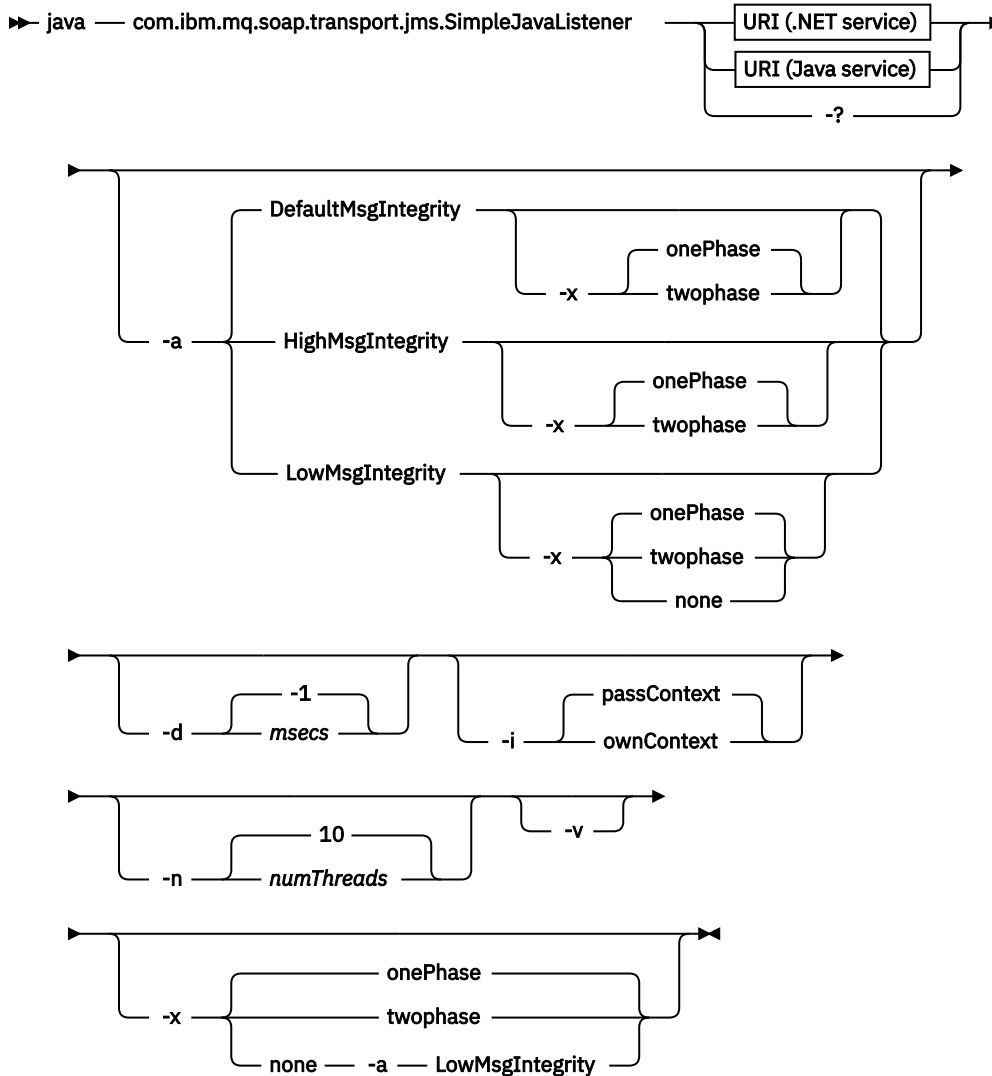
如果侦听器生成报告消息但在发送报告的过程中失败，那么会将报告消息发送至 DLQ。确保您的 DLQ 处理程序正确处理这些消息。

如果尝试写入死信队列时发生错误，那么会将消息写入 WebSphere MQ 错误日志。侦听器是否继续处理更多消息取决于选择了哪些消息持久性和事务选项。如果侦听器以一阶段事务方式运行，并且正在处理非持久请求消息，那么将废弃原始消息。WebSphere MQ SOAP 侦听器将继续执行。如果请求消息是持久的，那么会将请求消息回退到请求队列并退出侦听器。将请求队列设置为禁止获取，以防止意外触发的重新启动。

Syntax diagram .NET



Java



必需参数

URI 平台

请参阅第 885 页的『Web Service 部署的 URI 语法和参数』。

-?

打印描述如何使用命令的帮助文本。

可选参数

-a integrityOption

integrityOption 指定 WebSphere MQ SOAP 侦听器的行为 (如果无法将失败请求消息放入死信队列)。
integrityOption 可以采用下列其中一个值:

DefaultMsgIntegrity

对于非持久消息, 侦听器将显示一条警告消息, 并在废弃原始消息的情况下继续执行。对于持久消息, 它会显示错误消息, 回退请求消息以使其保留在请求队列上并退出。如果省略了 `-a` 选项, 或者未指定 *integrityOption*, 那么 `DefaultMsgIntegrity` 适用。

LowMsgIntegrity

对于持久消息和非持久消息, 侦听器会显示警告并继续执行, 从而废弃该消息。

HighMsgIntegrity

对于持久消息和非持久消息, 侦听器会显示一条错误消息, 回退请求消息以使其保留在请求队列中并退出。

部署实用程序检查 **-x** 和 **-a** 标志的兼容性。如果指定了 **-x none**，那么必须指定 **-a LowMsgIntegrity**。如果标志不兼容，那么部署实用程序将退出并返回错误消息，并且未执行任何部署步骤。

-d 毫秒

msecs 指定在任何线程上接收到请求消息时，WebSphere MQ SOAP 侦听器保持活动的毫秒数。如果 *msecs* 设置为 **-1**，那么侦听器将无限期保持活动状态。

-i 上下文

上下文 指定侦听器是否传递身份上下文。上下文 采用以下值：

passContext

将原始请求消息的身份上下文设置到响应消息中。SOAP 侦听器检查其是否有权从请求队列保存上下文并将其传递到响应队列。当打开请求队列以保存上下文和响应队列以传递上下文时，它会在运行时进行检查。如果它没有必需的权限，或者 **MQOPEN** 调用失败，并且未处理响应消息。响应消息将与包含来自失败 **MQOPEN** 的返回码的死信头一起放在死信队列上。然后，侦听器继续正常处理后续入局消息。

ownContext

SOAP 侦听器未传递上下文。返回的上下文反映用于运行侦听器的用户标识，而不是用于创建原始请求消息的用户标识。

源上下文中的字段由队列管理器设置，而不是由 SOAP 侦听器设置。

-n numThreads

numThreads 指定为 WebSphere MQ SOAP 侦听器生成的启动脚本中的线程数。缺省值为 **10**。如果消息吞吐量较高，请考虑增大此数字。

-v

-v 设置来自外部命令的详细输出。始终显示错误消息。使用 **-v** 输出可定制以创建定制部署脚本的命令。

-w serviceDirectory

serviceDirectory 是包含 Web Service 的目录。

-x 事务性

事务性 指定侦听器的事务控制类型。可以将 事务性 设置为下列其中一个值：

onePhase

使用 IBM WebSphere MQ 一阶段支持。如果系统在处理期间失败，那么会将请求消息重新传递到应用程序。WebSphere MQ 事务确保响应消息只写入一次。

twoPhase

使用两阶段支持。如果适当地编写服务，那么将在服务的单个已落实执行中，与其他资源协调，仅传递一次消息。此选项仅适用于服务器绑定连接。

none

无事务支持。如果系统在处理期间失败，那么请求消息可能会丢失，即使是持久消息也是如此。该服务可能已执行，也可能未执行，并且可能已写入或未写入响应，报告或死信消息。

部署实用程序检查 **-x** 和 **-a** 标志的兼容性。请参阅 **-a** 标志的描述以获取详细信息。

.NET 示例

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.Nojndi"
-w C:/wmqsoap/demos
-n 20
```

Java 示例

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi"
-n 20
```

针对 SOAP 发送方的 IBM WebSphere MQ 传输

为 Axis 和 .NET Framework 1 以及 .NET Framework 2 提供了发送方类。发送方构造 SOAP 请求并将其放在队列上，然后阻塞，直到从响应队列读取响应为止。您可以通过从 SOAP 客户机传递不同的 URI 来改变类的行为。对于 .NET Framework 3，请将 WebSphere MQ 定制通道用于 Microsoft Windows Communication Foundation (WCF)。

用途

WebSphere MQ SOAP 发送方将调用 Web Service 的 SOAP 请求放入 WebSphere MQ 请求队列。发送方根据 URI 中指定的选项或缺省值在 **MQRFH2** 头中设置字段。

如果您需要在使用 URI 选项的可能范围之外更改发件人的行为，请编写您自己的发件人。发送方可以使用 IBM WebSphere MQ Transport for SOAP 侦听器，也可以使用其他 SOAP 环境。发送方必须以 WebSphere MQ 定义的格式构造 SOAP 消息。此格式受 IBM WebSphere MQ SOAP 侦听器以及 WebSphere Application Server 和 CICS 提供的 SOAP 侦听器支持。发件人必须遵循 IBM WebSphere MQ 请求者的规则。IBM WebSphere MQ SOAP 侦听器返回响应和报告消息。有关如何在 **MQMD** 中设置报告选项的详细信息，请参阅第 865 页的『MQMD SOAP 设置』。报告选项控制 WebSphere MQ SOAP 侦听器返回的报告消息。

描述

WebSphere MQ SOAP Java 发送方已向 Axis 主机环境注册 `jms:URI` 前缀。发送方在类 `com.ibm.mq.soap.transport.jms.WMQSender` 中实现，该类派生自 `org.apache.axis.handlers.BasicHandler`。如果 Axis 主机环境检测到 `jms:URI` 前缀，那么它将调用 `com.ibm.mq.soap.transport.jms.WMQSender` 类。类在放置消息后阻塞，直到它从响应队列中读取响应为止。如果在超时时间间隔内未接收到响应，那么发送方将抛出异常。如果在超时时间间隔内接收到响应，那么将使用 Axis 框架向客户机返回响应消息。客户机应用程序必须能够处理这些响应消息。

对于 Microsoft .NET Framework 1 和 .NET Framework 2 服务，WebSphere MQ SOAP 发送方在类 `IBM.WMQSOAP.MQWebRequest` 中实现，该类派生自 `System.Net.WebRequest` 和 `System.Net.IWebRequestCreate`。如果 .NET Framework 1 或 .NET Framework 2 检测到 `jms:URI` 前缀，那么它将调用 `IBM.WMQSOAP.MQWebRequest` 类。发送方创建 `MQWebResponse` 对象以从响应队列读取响应消息并将其返回到客户机。

`com.ibm.mq.soap.transport.jms.WMQSender` 是最终类，`IBM.WMQSOAP.MQWebRequest` 已密封。不能通过创建子类来修改其行为。

参数

设置 URI 以控制 Web Service SOAP 客户机中 IBM WebSphere MQ SOAP 发送方的行为。部署实用程序将创建包含提供给部署实用程序的 URI 选项的 Web Service 客户机存根。

将通道定义表与 WebSphere MQ SOAP Transport for SOAP 发送方配合使用

客户机连接通道定义是在 Web Service URI 的 `ConnectionFactory` 属性中设置连接属性的替代方法。连接属性为 `clientChannel`，`clientConnection` 和 `SSL` 参数。

描述

通过定义客户机连接来创建客户机通道描述表。即使 Web Service 客户机连接到不同的队列管理器，也请在单个队列管理器上的连接表中创建所有连接。连接表的缺省名称和位置为 `queue manager directory/@ipcc/AMQCLCHL.TAB`。

通过设置 `com.ibm.mq.soap.transport.jms.mqchlurl` 系统属性将连接表的位置传递到 Java 客户机。

通过设置 `MQCHLLIB` 和 `MQCHLTAB` 环境变量，将连接表的位置传递到 .NET 客户机。

您可以在 Web Service URI 的 `ConnectionFactory` 属性中同时提供通道连接表和通道连接参数。`ConnectionFactory` 中设置的值优先于通道定义表中的值。

在 Java 中使用通道定义表

```
java -Dcom.ibm.msg.client.config.location=file:/C:/mydir/myjms.config MyAppClass
```

图 22: 使用配置文件启动 Java 客户机

```
com.ibm.mq.soap.transport.jms.mqchlurl=file:/C:/ibm/wmq/qmgrs/QM1/@ipcc/AMQCLCHL.TAB
```

图 23: *myjms.config*

事务

在启动侦听器时使用 `-x` 选项，以事务方式运行 Web Service。通过在服务 URI 中设置 `persistence` 选项来选择消息的完整性。

Web Service

在启动侦听器时使用 `-x` 选项，以事务方式运行 Web Service。在 .NET Framework 1 和 2 上，SOAP 侦听器使用 Microsoft 事务协调程序 (MTS)。在轴 1.4 上，SOAP 侦听器使用队列管理器协调事务。

Web Service 客户机

SOAP 发送方不是事务性的。

WebSphere MQ 绑定

您可以设置 SOAP 发送方的绑定类型。它可以作为 WebSphere MQ 服务器应用程序或客户机应用程序进行连接。您还可以将 SOAP 发送方作为 .NET 上的 XA 客户机进行绑定。

消息持久性

通过在 URI 中设置 `持久性` 选项来选择持久性级别。

Web Service 事务

您可以使用 Web Service 事务，因为 SOAP 发送方不是事务性的。如果您编写自己的 SOAP 发送方，并且打算使用 Web Service 事务，请不要创建事务性 SOAP 发送方。不能在单一事务中发送请求消息和接收应答消息。Web Service 事务不得协调发送和接收。

Web Service 部署的 URI 语法和参数

用于部署 IBM WebSphere MQ Web Service 的语法和参数在 URI 中定义。部署实用程序根据 Web Service 的名称生成缺省 URI。您可以通过将自己的 URI 定义为部署实用程序的参数来覆盖缺省值。部署实用程序将 URI 合并到生成的 Web Service 客户机存根中。

用途

使用统一资源标识 (URI) 指定 Web Service。语法图指定 IBM WebSphere MQ Transport for SOAP 中支持的 URI。URI 控制用于访问目标服务的特定于 IBM WebSphere MQ 的 SOAP 参数和选项。URI 与 .NET，Apache Axis 1，WebSphere Application Server CICS 所托管的 Web Service 兼容。

描述

该 URI 将合并到由部署实用程序生成的 Web Service 客户机类中。客户机将 URI 传递到 IBM WebSphere MQ 消息中的 IBM WebSphere MQ SOAP 发送方。URI 控制由 IBM WebSphere MQ SOAP 发送方和 IBM WebSphere MQ SOAP 侦听器执行的处理。

Syntax

The URI syntax is as follows:

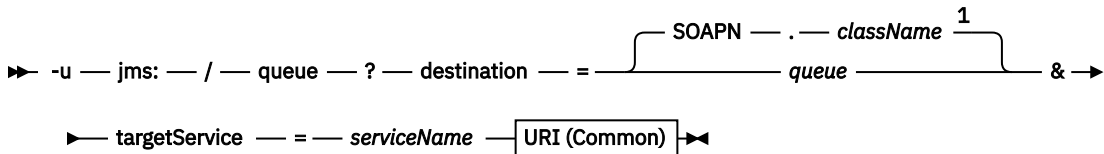
```
jms:/queue?name=value&name=value...
```

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 (&).

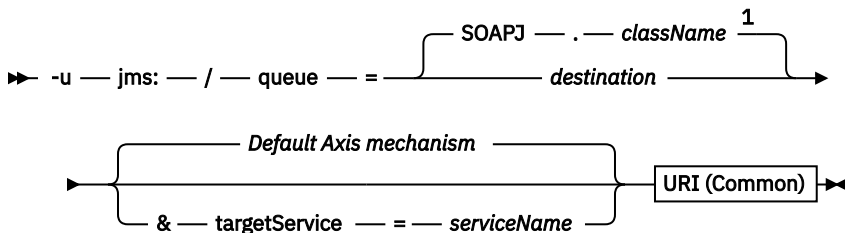
Parameter names are case-sensitive, as are names of IBM WebSphere MQ objects. If any parameter is specified more than once, the final occurrence of the parameter takes effect. Client applications can override a generated parameter by appending another copy of the parameter to the URI. If any additional 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.

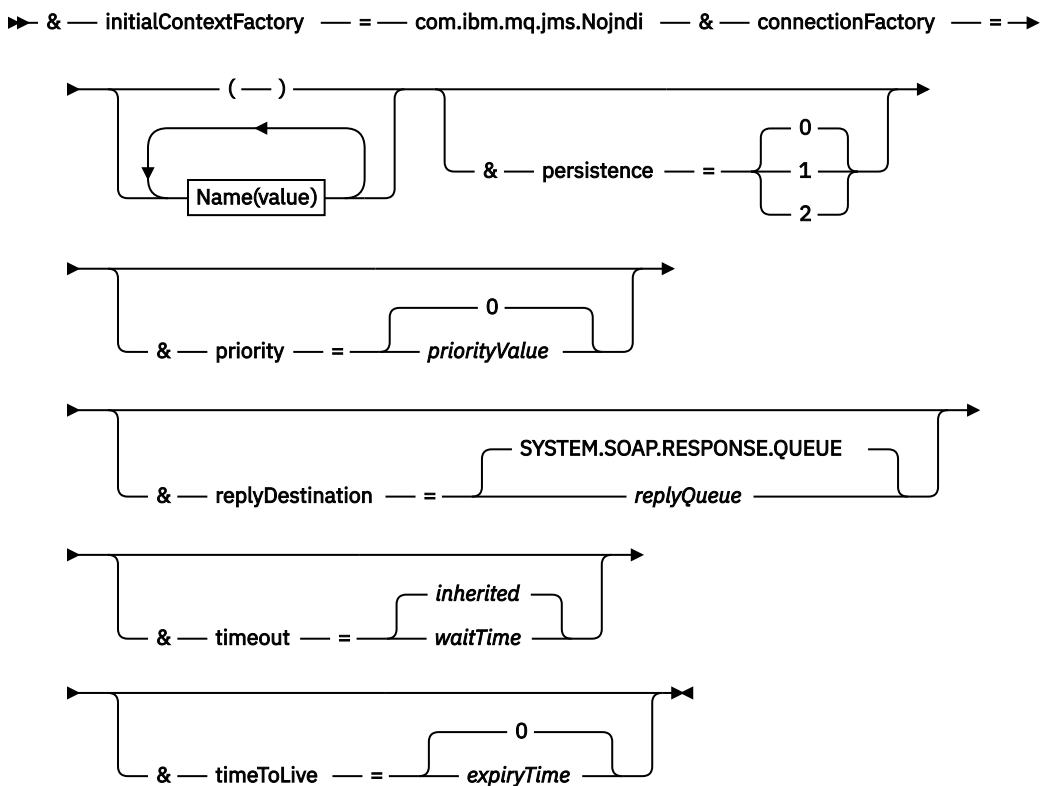
Syntax diagram
URI (.NET service)



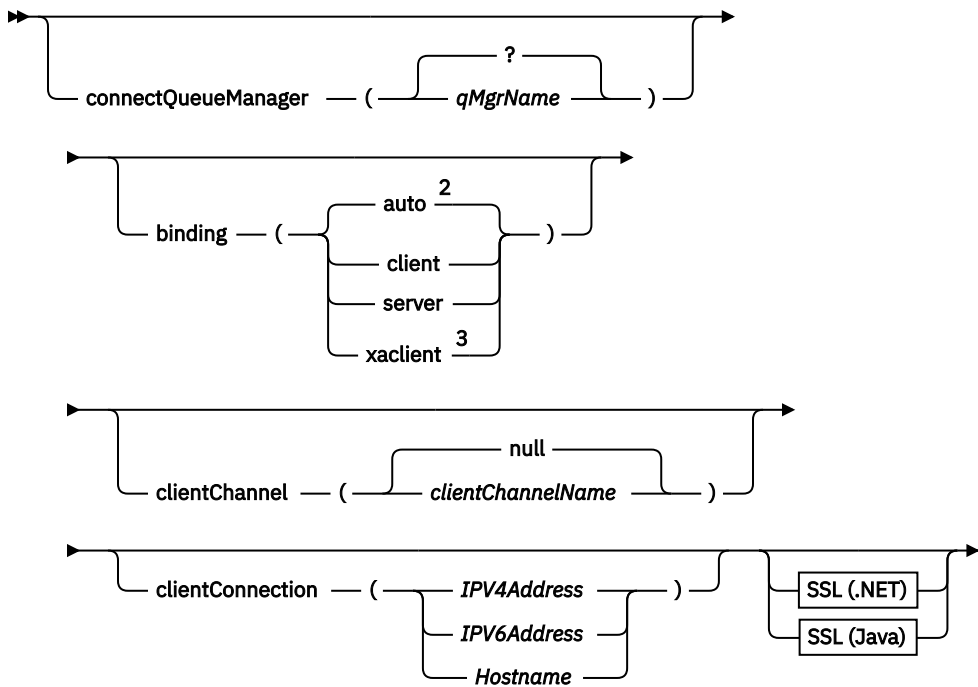
URI (Java service)



URI (Common)



Name(value)



注:

¹ The queue manager transforms *className* to a queue name following the steps described in [第 887 页的『目标到队列名称变换』](#)

² *client* is the default if other options appropriate for a client are specified; for example *clientConnection*.

³ *xaclient* applies to .NET only

目标到队列名称变换

1. *className* 以 SOAPJ. for Java 服务或 SOAPN. for .NET 服务为前缀。
2. 将从 *className* 参数中给定的完整路径名中除去文件扩展名。
3. 生成的字符串将截断为不超过 48 个字符
4. 目录分隔符将替换为句点字符。
5. 嵌入的空格将替换为下划线字符。
6. 驱动器前缀字母后面的冒号将替换为 .NET 服务的句点。

注: 在某些环境中, 部署实用程序生成的队列名称可能不唯一。部署实用程序将检查是否创建队列。您可以选择通过重构部署目录层次结构或定制提供的部署过程来覆盖部署实用程序。

必需的 URI 参数

destination=队列

queue 是请求目标的名称。它可以是队列或队列别名。如果是队列别名, 那么该别名可以解析为主题。

- 如果省略了 *-u* 参数, 那么将使用 [第 887 页的『目标到队列名称变换』](#) 中描述的步骤从 *classname* 生成 *queue*。
- 如果指定了 *-u* 参数, 那么 *queue* 是必需的, 并且必须是初始 *.jms:/queue* 之后 URI 的第一个参数? 字符串。指定 IBM WebSphere MQ 队列名称或通过 @ 符号连接的队列名称和队列管理器名称, 例如 *SOAPN.trandemos@WMQSOAP.DEMO.QM*。
- 部署实用程序将检查生成或提供的队列名称是否与现有队列的名称匹配。 [第 888 页的表 587](#) 中描述了所执行的操作。

| 表 587: 队列 验证 | | | |
|-------------------------------------|---|--|---|
| 是否存在侦听器脚本? | 侦听器脚本存在于 <code>./generated/server</code> 目录中 | | 侦听器脚本在 <code>./generated/server</code> 目录中不存在 |
| 侦听器脚本中的队列是否与 <code>queue</code> 匹配? | <code>queue</code> 与侦听器脚本中使用的请求队列不匹配 | <code>queue</code> 与侦听器脚本中正在使用的请求队列相匹配 | |
| 队列 存在 | <ul style="list-style-type: none"> - 部署存在错误。 - 该服务已部署在 <code>./generated/server</code> 中, 但正在使用另一个队列。 | <ul style="list-style-type: none"> - 部署正常继续。 - 该服务已部署在 <code>./generated/server</code> 中 | <ul style="list-style-type: none"> - 部署存在错误。 - 在 <code>./generated/server</code> 中找不到侦听器启动脚本, 但其他服务或应用程序正在使用队列。 |
| 队列 不存在 | | <ul style="list-style-type: none"> - 部署继续, 但存在警告。 - 先前部署可能已失败, 因为启动有效, 但缺少 <code>queue</code>。 | <ul style="list-style-type: none"> - 部署正常继续。 - 未从此目录部署任何服务。 |

&connectionFactory=名称 (值)

Name 是下列其中一个参数:

- [connectQueueManager \(qMgr 名称\)](#)
- [绑定 \(bindingType\)](#)
- [clientChannel\(通道\)](#)
- [clientConnection\(连接\)](#)
- [第 876 页的『必需的 SSL 参数 \(Java\)』](#)

请参阅 [第 889 页的『连接工厂参数』](#) 以获取这些参数的值的描述。

&targetService=serviceName

⁸ 在 .NET 上, *serviceName* 是位于部署目录中的 .NET 服务的名称, 例如:

`targetService=myService.asmx`。在 .NET 环境中, `targetService` 参数使单个 WebSphere MQ SOAP 侦听器能够处理针对多个服务的请求。必须从同一目录部署这些服务。

可选 URI 参数

&initialContextFactory=contextFactory

contextFactory 是必需的, 必须设置为 `com.ibm.mq.jms.Nojndi`。确保 `Nojndi.jar` 位于 WebSphere Application Server Web Service 客户机的类路径中。`Nojndi.jar` 根据 `connectionFactory` 和 `destination` 参数的内容而不是通过引用目录返回 Java 对象。

&targetService=serviceName

⁹ 在 "轴" 上, *serviceName* 是 Java 服务的标准名称, 例如:

`targetService=javaDemos.service.StockQuoteAxis`。如果未指定 `targetService`, 那么将使用缺省 Axis 机制装入服务。

&persistence=messagePersistence

messagePersistence 采用下列其中一个值:

0

从队列定义继承持久性。

⁸ 仅 .NET 服务

⁹ 仅限 Java 服务

- 1 该消息是非持久消息。
- 2 消息是持久的

&priority=priorityValue

priorityValue 在 0-9 范围内。0 是低优先级。缺省值为特定于环境的值，在 IBM WebSphere MQ 的情况下为 0。

&replyDestination=replyTo 队列

要用于响应消息的客户机端的队列。缺省应答队列为 SYSTEM.SOAP.RESPONSE.QUEUE。

- 运行 setupWMQSOAP 脚本以创建缺省 WebSphere MQ SOAP 对象。
- 为 *replyTo* 队列指定模型队列，以创建临时或永久动态响应队列。对于临时和永久动态响应队列，将为每个请求创建单独的动态队列实例。如果发生以下任何事件，那么将删除队列：
 - 响应已到达并已处理。
 - 请求超时。
 - 请求程序终止。

为了获得最佳性能，请使用临时动态队列而不是永久动态队列。请勿将持久请求消息发送到具有临时动态队列的 URI。IBM WebSphere MQ 侦听器 SOAP 无法处理消息并输出错误。客户机等待应答时超时。

- setupWMQSOAP 脚本创建名为 SYSTEM.SOAP.MODEL.RESPONSE.QUEUE 的缺省永久动态模型队列。

&timeout=waitTime

客户机等待响应消息的时间 (以毫秒计)。 *waitTime* 覆盖基础结构或客户机应用程序设置的值。如果未指定，那么将继承应用程序值 (如果已指定) 或基础结构缺省值。

注: 超时与 *timeToLive* 之间未强制实施任何关系。

&timeToLive=expiryTime

expiryTime 是消息到期之前的时间 (以毫秒为单位)。缺省值为零，这指示生存期不受限制。

注: 超时与 *timeToLive* 之间未强制实施任何关系。

连接工厂参数

connectQueueManager (qMgrName)

qMgrName 指定客户机连接到的队列管理器。缺省值为空白。

binding (bindingType)

bindingType 指定客户机如何连接到 *qMgrName*。缺省值为 auto。 *bindingType* 采用以下值:

auto

发送方按顺序尝试以下连接类型:

1. 如果指定了适合于客户机连接的其他选项，那么发送方将使用客户机绑定。其他选项为 *clientConnection* 或 *clientChannel*。
2. 使用服务器连接。
3. 使用客户机连接。

如果 SOAP 客户机上没有本地队列管理器，请在 *URI* 中使用 *binding(auto)*。为 SOAP 客户机构建客户机连接。

client

使用 *URI* 中的 *binding(client)* 为 SOAP 发送方构建客户机配置。

server

使用 *URI* 中的 *binding(服务器)* 为 SOAP 发送方构建服务器配置。如果连接具有客户机类型参数，那么连接将失败，并且 IBM WebSphere MQ SOAP 发送方将显示错误。客户机类型参数为 *clientConnection*， *clientChannel* 或 *SSL* 参数。

xaclient

xaclient 仅适用于 .NET，而不适用于 Java 客户机。使用 XA 客户机连接。

clientChannel(通道)

SOAP 客户机使用 通道 来建立 IBM WebSphere MQ 客户机连接。除非在服务器上启用了通道自动定义，否则 *channel* 必须与服务器连接通道的名称匹配。clientChannel 是必需参数，除非您提供了 "客户机连接定义" 表 (CCDT)。

通过设置 `com.ibm.mq.soap.transport.jms.mqchlurl` 提供 Java 中的 CCDT。在 .NET 中，设置 MQCHLLIB 和 MQCHLTAB 环境变量；请参阅 [第 884 页的『将通道定义表与 WebSphere MQ SOAP Transport for SOAP 发送方配合使用』](#)。

clientConnection(连接)

SOAP 客户机使用 连接 来建立 IBM WebSphere MQ 客户机连接。缺省主机名为 localhost，缺省端口为 1414。如果 *connection* 是 TCP/IP 地址，那么它采用三种格式之一，并且可以使用端口号作为后缀。

JMS 客户机可以使用格式：`hostname:port` 或使用格式 `%X` 对方括号进行 "转义"，其中 X 是表示 URI 的代码页中的方括号字符的十六进制值。例如，在 ASCII 中，`%28` 和 `%29` 分别用于 (和)。

.Net 客户机可以显式使用方括号：`hostname(port)` 或使用 "转义" 格式。

IPv4 地址

例如，`192.0.2.0`。

IPv6 地址

例如，`2001:DB8:0:0:0:0:0:0`。

主机名

例如，`www.example.com%281687%29`、`www.example.com:1687` 或 `www.example.com(1687)`。

SSL 平台

请参阅 [第 876 页的『必需的 SSL 参数 \(Java\)』](#)

样本 URI

注：

1. URI 中的 & 编码为 `&`；
2. 先前列示的所有参数都适用于客户机。
3. 只有 **destination**，**connectionFactory** 和 **initialContextFactory** 适用于 WCF 服务。

```
jms:/queue?  
destination=myQ&amp;connectionFactory=()&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

图 24: 用于 Axis 服务的 URI，仅提供必需参数

```
jms:/queue?destination=myQ&amp;connectionFactory=()&amp;targetService=MyService.asmx  
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

图 25: .NET 服务的 URI，仅提供必需参数

```
jms:/queue?destination=myQ@myRQM&amp;connectionFactory=connectQueueManager(myconnQM)  
binding(client)clientChannel(myChannel)clientConnection(myConnection)  
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

图 26: 用于 Axis 服务的 URI，提供一些可选 *connectionFactory* 参数

```
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
```

图 27: Axis 服务的 URI，提供 *connectionFactory* 参数的 *sslPeerName* 选项

Nojndi 机制

Nojndi 机制使使用 JNDI 接口的 JMS 程序能够使用与不使用 JNDI 的 WebSphere MQ 程序相同的 URI。

您可以使用 WebSphere MQ Transport for SOAP 在 WebSphere Application Server 上调用 Web Service。WebSphere Application Server SOAP over JMS 使用 JNDI 查找 JMS 资源。Web Service 客户机可能正在 .NET 上运行，或者使用 Axis 1.4 来调用 Web Service，而不使用 JNDI。要对客户机和服务器使用相同的 URL，它必须提供相同的信息，无论环境是否正在使用 JNDI。

Web Service 客户机传递到 WebSphere MQ Transport for SOAP 的 URI 包含特定的 WebSphere MQ 队列管理器和队列名称。这些名称由 WebSphere MQ SOAP 支持直接解析和使用。

Nojndi 机制将 JMS 程序使用的 `initialContextFactory` 定向到 `com.ibm.mq.jms.Nojndi`。`com.ibm.mq.jms.Nojndi` 类是 JNDI 接口的实现，用于从 URL 作为 `ConnectionFactory` 和 `Queue Java` 对象返回 `connectionFactory` 和 `destination`。如果 JMS 实现为 WebSphere MQ，那么 `MQConnectionFactory` 和 `MQQueue` 将从 `ConnectionFactory` 和 `Queue` 类继承。

通过使用 Nojndi 机制，您可以使用相同的 URL 向 WebSphere Application Server 和 .NET 提供相同的连接信息。

WebSphere MQ Axis 2 客户机的 W3C SOAP over JMS URI

定义 W3C SOAP over JMS URI，以使用 WebSphere MQ JMS 作为 SOAP 传输从 Axis 2 客户机调用 Web Service。Web Service 必须由支持 WebSphere MQ JMS 的服务器以及 SOAP/JMS 绑定的 W3C SOAP over JMS 候选项建议提供。

描述

W3C 候选值建议定义了 SOAP over JMS 绑定；[SOAP over Java Message Service 1.0](#)。URI Scheme for Java(tm) Message Service 1.0 对其示例也很有用。¹⁰

使用语法图来创建语法正确且被 WebSphere MQ Axis 2 客户机接受的 W3C SOAP over JMS URI。它仅限于定义 WebSphere MQ Axis 2 客户机接受的 URI。它是 W3C 建议的两个方面的子集：

1. `.jms-`变体 主题 不受支持，并且不得在传递到 WebSphere MQ Axis 2 客户机的 URI 中指定此主题。
2. 语法图中省略了以下属性，因为它们 JMS 属性，而不是 URI 的一部分。
 - a. `bindingVersion`
 - b. `contentType`
 - c. `soapAction`
 - d. `requestURI`
 - e. `isFault`

JMS 属性由 Axis 2 客户机或服务器设置。

该图通过定义定制参数 `connectionFactory` 来扩展 W3C 建议。`connectionFactory` 用作 JNDI 的替代方法，以指定 Axis 2 客户机如何使用队列连接到队列管理器。

WebSphere MQ Axis 2 客户机仅接受属性作为客户机应用程序传递给客户机的 URI 的一部分或作为环境变量。WebSphere MQ Axis 2 客户机无法处理 WSDL 文档。客户机应用程序或开发工具可能会处理 WSDL 并创建要传递到 Axis 2 客户机的 URI。WebSphere MQ Axis 2 客户机应用程序无法直接设置 JMS 消息属性。

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
```

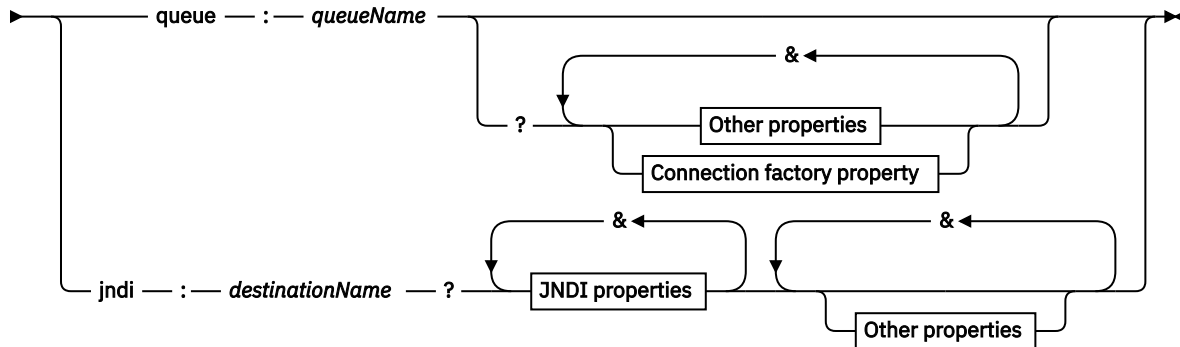
¹⁰ 在 W3C 规范参考中查找 [URI Scheme for JMS](#)，以获取最新草案。

If a parameter is set using an environment variable it overrides the value set in the URI.

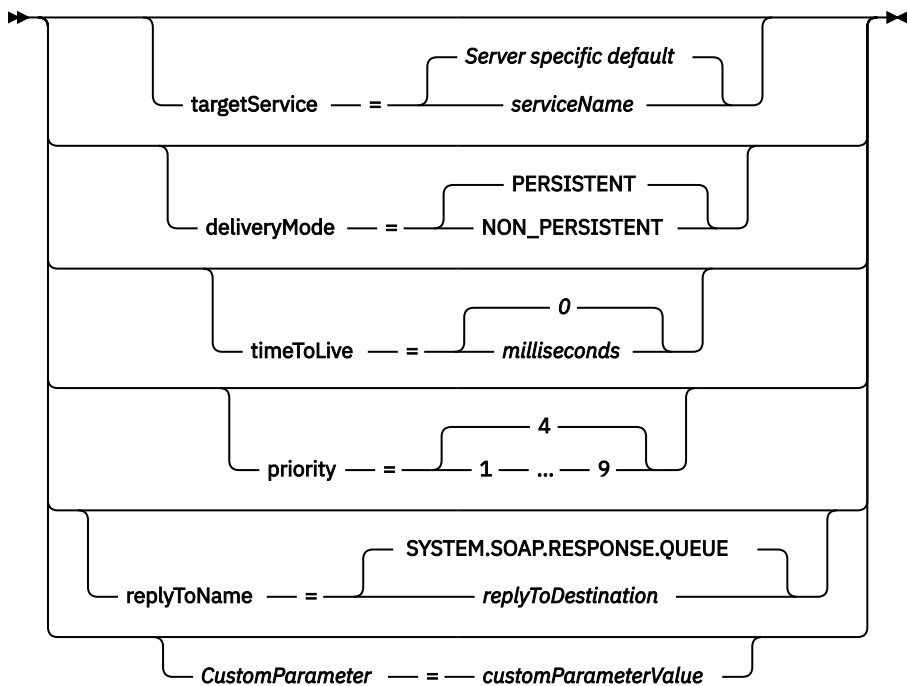
In accordance with the W3C recommendation, all the parameters can be repeated. The last instance of a parameter is used, unless overridden by an environment variable.

jms-uri

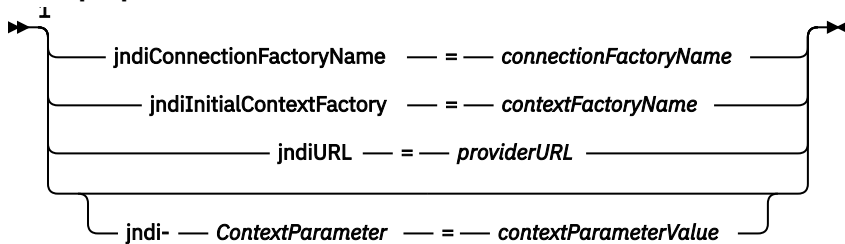
➔ jms: ➔



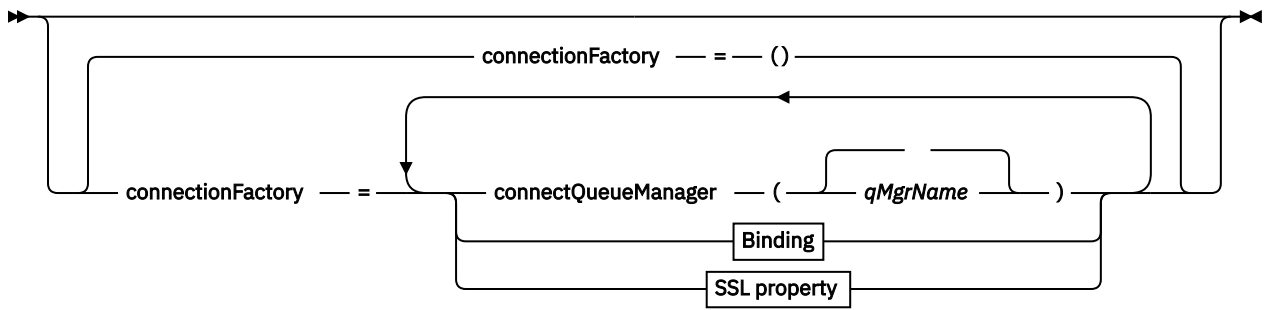
Other properties



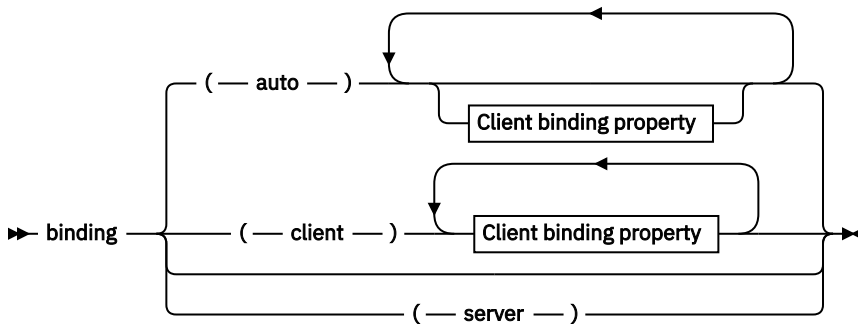
JNDI properties



Connection factory property

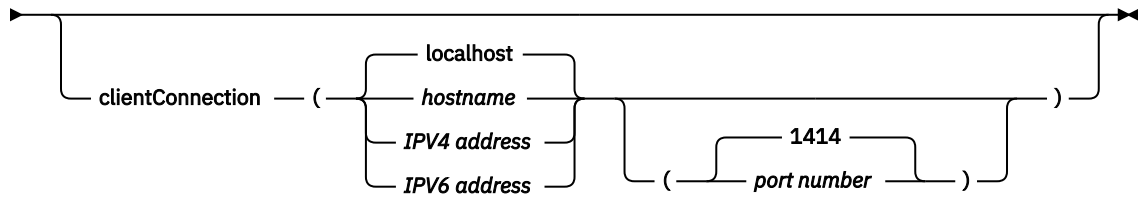


Binding



Client binding property

clientChannel — (— channel —) →



SSL property

sslCipherSuite — (— cipherSuite —)
 sslPeerName — (— peerName —)

sslKeyStore — (— KeyStoreName —) — sslKeyStorePassword — (— KeyStorePassword —)

sslTrustStore — (— TrustStoreName —) — sslTrustPassword — (— TrustStorePassword —)

sslKeyResetCount — (— byteCount —)

sslFipsRequired — (— fipsCertified —)

sslLDAPCRLServers — (— LDAPServerList —)

注:

¹ **jndiConnectionFactoryName**, **jndiConnectionName** and **jndiURL** are all required parameters. **jndi-ContextParameter** is optional.

参数

connectionFactory=connectionFactoryParameterList

connectionFactoryParameterList 是限定当目标变体为 *queue* 时 Axis 2 客户机如何连接到队列管理器的参数。

connectionFactory 不得与 *jndi* 目标变体一起指定。

这些参数不会传递到请求 URI 中的服务器。

如果省略了 *connectionFactory*，那么队列必须属于与 Axis 2 客户机在同一服务器上运行的缺省队列管理器。

connectionFactoryParameterList:

binding(bindingType)

bindingType 指定客户机如何连接到 *qMgrName*。缺省值为 *auto*。*bindingType* 采用以下值:

auto

发送方按顺序尝试以下连接类型:

1. 如果指定了适合于客户机连接的其他选项，那么发送方将使用客户机绑定。其他选项为 *clientConnection* 或 *clientChannel*。
2. 使用服务器连接。
3. 使用客户机连接。

如果 SOAP 客户机上没有本地队列管理器，请在 *URI* 中使用 *binding(auto)*。为 SOAP 客户机构建客户机连接。

client

使用 *URI* 中的 *binding(client)* 为 SOAP 发送方构建客户机配置。

server

使用 *URI* 中的 *binding(服务器)* 为 SOAP 发送方构建服务器配置。如果连接具有客户机类型参数，那么连接将失败，并且 IBM WebSphere MQ SOAP 发送方将显示错误。客户机类型参数为 *clientConnection*，*clientChannel* 或 *SSL* 参数。

xaclient

xaclient 仅适用于 .NET，而不适用于 Java 客户机。使用 XA 客户机连接。

clientChannel(通道)

SOAP 客户机使用 *通道* 来建立 IBM WebSphere MQ 客户机连接。除非在服务器上启用了通道自动定义，否则 *channel* 必须与服务器连接通道的名称匹配。*clientChannel* 是必需参数，除非您提供了 "客户机连接定义" 表 (CCDT)。

通过设置 *com.ibm.mq.soap.transport.jms.mqchlurl* 提供 Java 中的 CCDT。在 .NET 中，设置 *MQCHLLIB* 和 *MQCHLTAB* 环境变量; 请参阅第 884 页的『将通道定义表与 WebSphere MQ SOAP Transport for SOAP 发送方配合使用』。

clientConnection(连接)

SOAP 客户机使用 *连接* 来建立 IBM WebSphere MQ 客户机连接。缺省主机名为 *localhost*，缺省端口为 *1414*。如果 *connection* 是 TCP/IP 地址，那么它采用三种格式之一，并且可以使用端口号作为后缀。

JMS 客户机可以使用格式: *hostname:port* 或使用格式 *%X* 对方括号进行 "转义"，其中 *X* 是表示 URI 的代码页中的方括号字符的十六进制值。例如，在 ASCII 中，*%28* 和 *%29* 分别用于 (和)。

.Net 客户机可以显式使用方括号: *hostname(port)* 或使用 "转义" 格式。

IPv4 地址

例如，*192.0.2.0*。

IPv6 地址

例如，*2001:DB8:0:0:0:0:0:0*。

主机名

例如，*www.example.com%281687%29*、*www.example.com:1687* 或 *www.example.com(1687)*。

sslCipherSuite (CipherSuite)

CipherSuite 指定通道上使用的 *sslCipherSuite*。客户机指定的 *CipherSuite* 必须与服务器连接通道上指定的 *CipherSuite* 相匹配。

sslFipsRequired (fipsCertified)

fipsCertified 指定 *CipherSpec* 还是 *CipherSuite* 必须在通道上的 IBM WebSphere MQ 中使用 FIPS 认证的密码术。设置 *fipsCertified* 的效果与在 MQCONN 调用上设置 MQSCO 结构的 FipsRequired 字段相同。

sslKeyStore (KeyStore 名称)

KeyStore 名称 指定通道上使用的 *sslKeyStoreName*。密钥库包含用于向服务器认证客户机的客户机专用密钥。如果 SSL 连接配置为接受匿名客户机连接，那么密钥库是可选的。

sslKeyResetCount (bytecount)

bytecount 指定在必须重新协商 SSL 密钥之前通过 SSL 通道传递的字节数。要禁用 SSL 密钥的重新协商，请省略该字段或将其设置为零。零是某些环境中唯一受支持的值，请参阅 [重新协商 WebSphere MQ classes for Java](#) 中的密钥。设置 *sslKeyResetCount* 的效果与设置 MQCONN 调用中的 MQSCO 结构中的 *KeyResetCount* 字段相同。

sslKeyStorePassword (KeyStore 密码)

KeyStore 密码 指定通道上使用的 *sslKeyStorePassword*。

sslLDAPCRLServers (LDAPServerList)

LDAPServerList 指定要用于证书撤销列表检查的 LDAP 服务器的列表。

对于启用 SSL 的客户机连接，*LDAPServerList* 是要用于证书撤销列表 (CRL) 检查的 LDAP 服务器的列表。将针对列出的某个 LDAP CRL 服务器检查队列管理器提供的证书；如果找到该证书，那么连接将失败。将依次尝试每个 LDAP 服务器，直到与其中一个服务器建立连接为止。如果无法连接到任何服务器，那么将拒绝该证书。成功建立与其中一个服务器的连接后，将根据该 LDAP 服务器上存在的 CRL 接受或拒绝该证书。

如果 *LDAPServerList* 为空，那么不会针对证书撤销列表检查属于队列管理器的证书。如果提供的 LDAP URI 列表无效，那么将显示错误消息。设置此字段的效果与包含 MQAIR 记录并从 MQCONN 上的 MQSCO 结构访问这些记录的效果相同。

sslPeerName (peerName)

peerName 指定通道上使用的 *sslPeer* 名称。

sslTrustStore (TrustStore 名称)

TrustStore 名称 指定通道上使用的 *sslTrustStoreName*。信任库保存服务器的公用证书或其密钥链，以向客户机认证服务器。如果使用认证中心的根证书来认证服务器，那么信任库是可选的。在 Java 中，根证书保存在 JRE 证书库 cacerts 中。

sslTrustStorePassword (TrustStore 密码)

TrustStorePassword 指定通道上使用的 *sslTrustStorePassword*。

CustomParameter=customParameter 值

CustomParameter 是用户定义的定制参数名称，*customParameterValue* 是参数的值。

Axis 2 客户机未使用的定制参数由 Axis 2 客户机发送到 SOAP 服务器。请参阅服务器文档。*connectionFactory* 是由 Axis 2 客户机使用的定制参数，不会传递到服务器。

CustomParameter 必须与现有参数的名称不匹配。

如果 *CustomParameter* 以字符串 *jndi-* 开头，那么它用于查找 JNDI 目标；请参阅 [jndi-](#)。

deliveryMode=deliveryMode

deliveryMode 设置消息持久性。缺省值为 PERSISTENT。

jndi:destinationName

destinationName 是映射到 JMS 队列的 JNDI 目标名称。如果指定了 *jndi* 目标变体，那么必须提供 *destinationName*。

jndiConnectionFactoryName=connectionFactory 名称

connectionFactoryName 设置连接工厂的 JNDI 名称。如果目标变体是 *jndi*，那么必须提供 *connectionFactoryName*。

jndiInitialContextFactory=contextFactory 名称

contextFactoryName 设置初始上下文工厂的 JNDI 名称。如果目标变体是 *jndi*，那么必须提供 *contextFactoryName*。请参阅 [使用 JNDI 来检索 JMS 应用程序中的受管对象](#)。

jndiURL=providerURL

jndiURL 设置 JNDI 提供程序的 URL 名称。如果目标变体是 *jndi*，那么必须指定 *jndiURL*。

jndi-ContextParameter=contextParameter 值

jndi-ContextParameter 是用于将信息传递到 JNDI 提供程序的定制参数的用户定义名称。*contextParameter* 值是传递的信息。

priority=priorityValue

priorityValue 设置 JMS 消息优先级。0 低，9 高。缺省值为 4。

queue:queueName

queueName 是放置 SOAP 请求的 JMS 队列的名称。如果指定了队列变体，那么必须提供队列名称。如果队列不属于客户机所在服务器上的缺省队列管理器，请设置 [connectionFactory](#) 参数。

replyToName=replyTo 目标

replyToDestination 设置目标队列名称。如果目标变体是 *jndi*，那么该名称是必须映射到队列的 JNDI 名称。如果变体是 *queue*，那么名称是 JMS 队列。缺省值为 `SYSTEM.SOAP.RESPONSE.QUEUE`。

targetService=serviceName

SOAP 服务器用于启动目标 Web Service 的名称。

在轴上，*serviceName* 是 Java 服务的标准名称，例如：

`targetService=www.example.org.StockQuote`。如果未指定 *targetService*，那么将使用缺省 Axis 机制装入服务。

timeToLive=毫秒

将 毫秒 设置为消息到期之前的时间。缺省值 0 是永不到期的消息。

示例

```
jms:jndi:REQUESTQ
?jndiURL=file:/C:/JMSAdmin
&jndiInitialContextFactory=com.sun.jndi.fscontext.RefFSContextFactory
&jndiConnectionFactoryName=ConnectionFactory
&replyToName=RESPONSEQ
&deliveryMode(NON_PERSISTENT)
```

图 28: 使用 *jms:jndi* 发送 SOAP/JMS 请求

```
jms:queue:SOAPJ.demos
?connectionFactory=connectQueueManager(QM1)
Bind(Client)
ClientChannel(SOAPClient)
ClientConnection(www.example.org(1418))
&deliveryMode(NON_PERSISTENT)
```

图 29: 使用 *jms:queue* 发送 SOAP/JMS 请求

受支持的 Web Service

无需修改已编写为作为 Web Service 运行的代码即可使用 IBM WebSphere MQ Transport for SOAP。您确实需要以不同方式部署服务，以便与 IBM WebSphere MQ Transport for SOAP 一起运行，而不是使用 HTTP。

描述

WebSphere MQ Transport for SOAP 提供了一个 SOAP 侦听器，用于运行 .NET Framework 1 和 .NET 2 以及 Axis 1.4 的服务。用于 Microsoft Windows Communication Foundation 的 WebSphere MQ 定制通道运行 .NET Framework 3 的服务。WebSphere Application Server 和 CICS 支持通过 WebSphere MQ Transport for SOAP 运行服务。创建定制导出以使用 WebSphere Enterprise Service Bus 或 WebSphere Process Server。

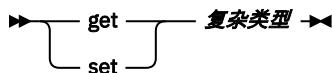
WebSphere MQ SOAP 侦听器可以通过事务方式处理 SOAP 请求。使用 `-x` 选项运行 **amqwdeployWMQService**。只有使用服务器绑定的侦听器才支持两阶段选项。其他环境可能为 WebSphere MQ Transport for SOAP 提供事务性支持。请查阅他们的文档。

WebSphere MQ Transport for SOAP 当前不支持已提交到 W3C 的新兴业界标准 SOAP over JMS 协议。您可以通过查找 JMS BindingVersion 属性来区分写入新标准的 SOAP/JMS 消息。WebSphere MQ Transport for SOAP 未设置 BindingVersion 属性。

Axis 1.4

通常可以在不进行修改的情况下使用 Java 类。轴引擎必须支持 Web Service 中方法的任何自变量的类型。请参阅 Axis 文档以获取更多详细信息。如果服务使用复杂对象作为自变量，或者返回一个对象，那么该对象必须符合 Java™ 规范。请参阅 [第 898 页的图 32](#)，[第 899 页的图 33](#) 和 [第 899 页的图 34](#) 中的示例：

1. 具有公共无参数构造函数。
2. 任何复杂类型的 bean 都必须具有以下格式的公用 getter 和 setter:



使用 **amqwdeployWMQService** 实用程序准备服务以进行部署。该服务由 WebSphere MQ SOAP 侦听器调用，该侦听器使用 `axis.jar` 来运行该服务。

Axis 1.4 支持的唯一两阶段事务管理器是 WebSphere MQ。

提供的部署实用程序不支持服务将其他包中的对象返回到服务本身的情况。要使用另一个包中返回的对象，请编写您自己的部署实用程序。您可以将部署实用程序基于提供的样本，或者使用 `-v` 选项来捕获其生成的命令。修改命令以生成定制脚本。

如果服务使用 Axis 基础结构和 WebSphere MQ SOAP 运行时环境外部的类，那么必须设置正确的 CLASSPATH。要更改 CLASSPATH，请通过下列其中一种方法修改用于启动或定义侦听器以包含所需服务的生成脚本：

- 在调用 **amqwsetcp** 之后，直接修改脚本中的 CLASSPATH。
- 创建特定于服务的脚本以定制 CLASSPATH，并在调用 **amqwsetcp** 之后在生成的脚本中调用此脚本。
- 创建定制部署过程以自动定制生成的脚本中的 CLASSPATH。

.NET Framework 1 和 .NET Framework 2

已准备为 HTTP Web Service 的服务无需修改即可用作 WebSphere MQ Web Service。需要使用 **amqwdeployWMQService** 实用程序进行部署。

.NET Framework 1 和 .NET 2 支持的唯一两阶段事务管理器是 Microsoft Transaction Server (MTS)。

如果未准备服务代码作为 HTTP Web Service，那么必须将其转换为 Web Service。将类声明为 Web Service，并确定如何格式化每种方法的参数。您必须检查服务方法的任何自变量是否与环境兼容。[第 898 页的图 30](#) 并且 [第 898 页的图 31](#) 显示已准备为 Web Service 的 .NET 类。添加的内容以粗体显示。

[第 898 页的图 30](#) 将代码后编程模型用于 .NET Web Service。在代码后模型中，服务的源与 `.asmx` 文件分开。`.asmx` 文件使用 `Codebehind` 关键字声明关联源文件的名称。WebSphere MQ 具有内联和代码支持 .NET Web Service 的样本。

.NET Web Service 源必须在 **amqwdeployWMQService** 部署实用程序进行部署之前进行编译。该服务已编译到库 (`.dll`) 中。该库必须放在部署目录的 `./bin` 子目录中。

.NET Framework 3

为 Microsoft Windows Communication Foundation (WCF) 创建 WebSphere MQ 定制通道，以调用部署到 .NET Framework 3 的服务。请参阅 [IBM WebSphere MQ Microsoft Windows Communication Foundation \(WCF\) 的定制通道](#)，以获取有关如何配置 WCF 以使用 WebSphere MQ Transport for SOAP 的描述。

WebSphere Application Server

您可以使用 WebSphere MQ Transport for SOAP 来调用由 WebSphere Application Server 托管的 Web Service; 请参阅 [使用 SOAP over JMS 来传输 Web Service \(不推荐\)](#)。

您需要修改通过将 JMS 服务部署到 WebSphere Application Server 而生成的 WSDL, 以便生成 Web Service 客户机。通过部署到 WebSphere Application Server 创建的 WSDL 包含具有对 JMS InitialContextFactory 的 JNDI 引用的 URI。您需要修改对 Nojndi 的 JNDI 引用并提供连接属性, 如第 885 页的『Web Service 部署的 URI 语法和参数』中所述。

CICS

您可以使用 WebSphere MQ Transport for SOAP 来调用 CICS 应用程序; 请参阅 [为 Web Service 配置 CICS 系统](#)。

WebSphere Enterprise Service Bus 和 WebSphere Process Server for Multiplatforms

仅当使用缺省 WebSphere Application Server 消息传递提供程序时, WebSphere ESB 和 WebSphere Process Server for Multiplatforms 才支持具有就绪构建绑定的 SOAP over JMS。为 JMS 创建定制绑定以支持 WebSphere MQ Transport for SOAP。请参阅 [JMS 数据绑定 \(JMS data bindings\)](#)。

示例

```
<%@ WebService Language="C#" CodeBehind="Quote.aspx.cs" Class="Quote.QuoteDotNet" %>
```

图 30: .NET Framework 2 的服务定义: *Quote.aspx*

```
<%@ WebService Language="C#" CodeBehind="Quote.aspx.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();
        }
    }
}
```

图 31: .NET Framework 2 的服务实现: *Quote.aspx.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;
}
```

图 32: 使用复杂类型的 *Java JAX-RPC* 服务接口

```

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;
    }
}

```

图 33: 使用复杂类型的 *Java JAX-RPC* 服务实现

```

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; }
}

```

图 34: 复杂类型的 *Java JAX-RPC* 服务 *Bean* 实现

用于 SOAP Web Service 客户机的 IBM WebSphere MQ 传输

您可以将现有 SOAP over HTTP 客户机与 IBM WebSphere MQ Transport for SOAP 一起复用。您必须对代码和构建过程进行一些小的修改，以将客户机转换为使用 IBM WebSphere MQ Transport for SOAP。

代码

JAX-RPC 客户机必须以 Java 编写。 .NET Framework 1 和 2 客户机可以使用任何使用公共语言运行时的语言编写。 C# 和 Visual Basic 中提供了代码示例。

事务支持级别取决于客户机环境和 SOAP 交互模式。 SOAP 请求和 SOAP 应答不能是同一原子事务的一部分。

必须在 .NET Framework 1 .NET Framework 2 客户机中调用 `IBM.WMQSOAP.Register.Extension()`。在 JAX-RPC Java Web Service 客户机调用 `com.ibm.mq.soap.Register.extension` 以注册 WebSphere MQ SOAP 发送方。此方法将 WebSphere MQ Transport for SOAP 发送方注册为使用 `.jms:` 协议的 SOAP 消息处理程序。

要创建 .NET Framework 3 客户机，请使用 **svcutil** 工具生成 Windows Communication Foundation 客户机代理；请参阅 [使用 svcutil 工具以及来自正在运行的服务的元数据生成 WCF 客户机代理和应用程序配置文件](#)。

构建和运行 .NET Framework 1 和 2 客户机所需的库

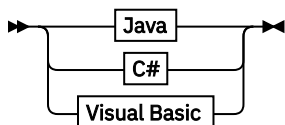
- 安加肥皂
- 系统
- System.Web.Services
- System.Xml

构建和运行 Axis 1.4 客户机所需的库

- `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\saa.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

►► `com.ibm.mq.soap.Register.extension()` ◄◄

C#

►► `IBM.WMQSOAP.Register.Extension();` ◄◄

Visual Basic

►► `IBM.WMQSOAP.Register.Extension` ◄◄

客户示例

第 901 页的图 35 是使用内联编程模型的 .NET Framework 1 或 .NET Framework 2 C# 客户机的示例。**IBM.WMQSOAP.Register.Extension()** 方法向 .NET 注册 WebSphere MQ SOAP 发送方作为 `jms:` 协议处理程序。

```

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);
            }
        }
    }
}

```

图 35: C# Web Service 客户机样本

第 901 页的图 36 是使用 JAX-RPC 静态代理客户机接口的 Java 客户机的示例。

com.ibm.mq.soap.Register.extension(); 方法向服务代理注册 WebSphere MQ SOAP 发送方以处理 jms: 协议。

```

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());
        }
    }
}

```

图 36: Java Web Service 客户机示例

用户出口、API 出口和可安装服务参考

使用本节中提供的链接来帮助您开发用户出口，API 出口和可安装服务应用程序：

- [第 902 页的『MQIEP 结构』](#)
- [第 905 页的『数据转换出口引用』](#)
- [第 908 页的『MQ_PUBLISH_EXIT - 发布出口』](#)
- [第 915 页的『通道出口调用和数据结构』](#)
- [第 972 页的『API 出口参考』](#)
- [第 1028 页的『可安装服务接口参考信息』](#)

相关概念

[第 7 页的『MQI 应用程序参考』](#)

使用本节中提供的链接来帮助您开发 MQI 应用程序：

[第 1277 页的『Java 库的 IBM WebSphere MQ 类』](#)

Java 库的 IBM WebSphere MQ 类的位置因平台而异。您启动应用程序时需指定此位置。

相关任务

[开发应用程序](#)

相关参考

[第 850 页的『SOAP 引用』](#)

[按字母顺序排列的 WebSphere MQ Transport for SOAP 参考信息。](#)

第 1087 页的『[IBM WebSphere MQ Bridge for HTTP 的参考材料](#)』
IBM WebSphere MQ Bridge for HTTP 的参考主题，按字母顺序排列

第 1120 页的『[IBM WebSphere MQ .NET 类和接口](#)』
IBM WebSphere MQ .NET 类和接口按字母顺序列出。描述了属性，方法和构造函数。

第 1176 页的『[IBM WebSphere MQ C++ 类](#)』
IBM WebSphere MQ C++ 类封装了 IBM WebSphere MQ 消息队列接口 (MQI)。有一个单独的 C++ 头文件 **imqi.hpp**，它涵盖所有这些类。

用于 JMS 的 WebSphere MQ 类

MQIEP 结构

MQIEP 结构包含允许出口进行的每个函数调用的入口点。

字段

StrucId

类型: MQCHAR4 -输入

结构标识。值如下所示:

MQIEP_STRUC_ID

版本

类型: MQLONG-输入

结构版本号。值如下所示:

MQIEP_VERSION_1

版本 1 结构版本号。

MQIEP_CURRENT_VERSION

结构的当前版本。

StrucLength

类型: MQLONG

MQIEP 结构的大小 (以字节计)。值如下所示:

MQIEP_LENGTH_1

标志

类型: MQLONG

提供有关函数地址的信息。用于指示库是否为线程库的标志可与用于指示库是客户机还是服务器库的标志一起使用。

以下值用于不指定库信息:

MQIEPF_NONE

下列其中一个值用于指定共享库是线程库还是非线程库:

MQIEPF_NON_THREADED_LIBRARY

非线程共享库

MQIEPF_THREADED_LIBRARY

线程共享库

下列其中一个值用于指定共享库是客户机还是服务器共享库:

MQIEPF_CLIENT_LIBRARY

客户机共享库

MQIEPF_LOCAL_LIBRARY

服务器共享库

已保留

类型:MQPTR

MQBACK_Call

类型:PMQ_BACK_CALL

MQBACK 调用的地址。

MQBEGIN_Call

类型:PMQ_BEGIN_CALL

MQBEGIN 调用的地址。

MQBUFMH_Call

类型:PMQ_BUFMH_CALL

MQBUFMH 调用的地址。

MQCB_Call

类型:PMQ_CB_CALL

MQCB 调用的地址。

MQCLOSE_Call

类型:PMQ_CLOSE_CALL

MQCLOSE 调用的地址。

MQCMIT_Call

类型:PMQ_CMIT_CALL

MQCMIT 调用的地址。

MQCONN_Call

类型:PMQ_CONN_CALL

MQCONN 调用的地址。

MQCONNX_Call

类型:PMQ_CONNX_CALL

MQCONNX 调用的地址。

MQCRTMH_Call

类型:PMQ_CRTMH_CALL

MQCRTMH 调用的地址。

MQCTL_Call

类型:PMQ_CTL_CALL

MQCTL 调用的地址。

MQDISC_Call

类型:PMQ_DISC_CALL

MQDISC 调用的地址。

MQDLTMH_Call

类型:PMQ_DLTMH_CALL

MQDLTMH 调用的地址。

MQDLTMP_Call

类型:PMQ_DLTMP_CALL

MQDLTMP 调用的地址。

MQGET_调用

类型:PMQ_GET_CALL

MQGET 调用的地址。

MQINQ_Call

类型 :PMQ_INQ_CALL

MQINQ 调用的地址。

MQINQMP_Call

类型 :PMQ_INQMP_CALL

MQINQMP 调用的地址。

MQMHBUF_Call

类型 :PMQ_MHBUF_CALL

MQMHBUF 调用的地址。

MQOPEN_Call

类型 :PMQ_OPEN_CALL

MQOPEN 调用的地址。

MQPUT_Call

类型 :PMQ_PUT_CALL

MQPUT 调用的地址。

MQPUT1_Call

类型 :PMQ_PUT1_CALL

MQPUT1 调用的地址。

MQSET_Call

类型 :PMQ_SET_CALL

MQSET 调用的地址。

MQSETMP_Call

类型 :PMQ_SETMP_CALL

MQSETMP 调用的地址。

MQSTAT_Call

类型 :PMQ_STAT_CALL

MQSTAT 调用的地址。

MQSUB_Call

类型 :PMQ_SUB_CALL

MQSUB 调用的地址。

MQSUBRQ_Call

类型 :PMQ_SUBRQ_CALL

MQSUBRQ 调用的地址。

MQXCNVC_Call

类型 :PMQ_XCNVC_CALL

MQXCNVC 调用的地址。

MQXCLWLN_Call

类型 :PMQ_XCLWLN_CALL

MQXCLWLN 调用的地址。

MQXDX_Call

类型 :PMQ_XDX_CALL

MQXDX 调用的地址。

MQXEP_Call

类型 :PMQ_XEP_CALL

MQXEP 调用的地址。

MQZEP_Call

类型:PMQ_ZEP_CALL

MQZEP 调用的地址。

C. 声明

```
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 */
};
```

数据转换出口引用

对于 z/OS，必须以汇编语言编写数据转换出口。对于其他平台，建议您使用 C 编程语言。

为了帮助您创建数据转换出口程序，提供了以下内容：

- 框架源文件
- 转换字符调用
- 用于创建对数据类型结构执行数据转换的代码片段的实用程序。此实用程序仅接受 C 输入。在 z/OS 上，它生成汇编程序代码。

有关编写程序的过程，请参阅：

- [在 UNIX and Linux 系统上为 WebSphere MQ 编写数据转换出口](#)
- [为 WebSphere MQ for Windows 编写数据转换出口](#)

框架源文件

这些可用作编写数据转换出口程序时的起点。

第 906 页的表 588 中列出了提供的文件。

表 588: 框架源文件

| 平台 | 文件 |
|---|--|
| AIX | amqsvfc0.c |
| IBM i | QMOMSAMP/QCSRC (AMQSVFC4) |
| HP-UX | amqsvfc0.c |
| Linux | amqsvfc0.c |
| z/OS | CSQ4BAX8 (第 906 页的『1』) CSQ4BAX9 (第 906 页的『2』) CSQ4CAX9 (第 906 页的『3』) |
| Solaris | amqsvfc0.c |
| 在 Windows 系统上 | amqsvfc0.c |
| 注意: <ol style="list-style-type: none"> 1. 说明 MQXCVNC 调用。 2. 实用程序生成的代码段的包装器, 用于除 CICS 以外的所有环境中。 3. 实用程序生成的用于 CICS 环境的代码片段的包装器。 | |

转换字符调用

使用数据转换出口程序中的 MQXCNVN (转换字符) 调用将字符消息数据从一个字符集转换为另一个字符集。对于某些多字节字符集 (例如, UCS2 字符集), 必须使用相应的选项。

不能从出口中执行其他 MQI 调用; 尝试执行此类调用失败, 原因码为 MQRC_CALL_IN_PROGRESS。请参阅第 802 页的『MQXCNVN-转换字符』, 以获取有关 MQXCNVN 调用和相应选项的更多信息。

创建转换出口代码的实用程序

使用此信息可了解有关创建转换出口代码的更多信息。

用于创建转换-退出代码的命令包括:

IBM i

CVTMQMDTA (转换 WebSphere MQ 数据类型)

Windows 和 UNIX and Linux 系统

crtmqcvx (创建 WebSphere MQ 转换-出口)

针对平台的命令生成一个代码片段, 用于在数据类型结构上执行数据转换, 以便在数据转换出口程序中使用。该命令采用包含一个或多个 C 语言结构定义的文件。

Windows UNIX and Linux 系统中的错误消息

crtmqcvx 命令返回 AMQ7953 到 AMQ7970 范围内的消息。

这些消息在 [原因码 WebSphere MQ](#) 消息中列出。

错误主要有两种类型:

- 处理无法继续时的主要错误, 例如语法错误。

屏幕上会显示一条消息，给出输入文件中错误的行号。输出文件可能已部分创建。

- 当显示一条消息，指出已找到问题但可继续对结构进行解析时，会发生其他错误。

已创建输出文件，其中包含有关已发生的问题的错误信息。此错误信息以 `#error` 为前缀，因此任何编译器都不会接受生成的代码，而不会进行干预以纠正问题。

有效语法

实用程序的输入文件必须符合 C 语言语法。

如果您不熟悉 C，请参阅本主题中的 [C 示例](#)。

此外，请注意以下规则：

- 仅在 `struct` 关键字之前识别 `typedef`。
- 结构声明上需要结构标记。
- 您可以使用空的方括号 `[]` 来表示消息末尾的可变长度数组或字符串。
- 不支持多维数组和字符串数组。
- 可识别以下其他数据类型：

- `MQBOOL`
- `MQBYTE`
- `MQCHAR`
- `MQFLOAT32`
- `MQFLOAT64`
- `MQSHORT`
- `MQLONG`
- `MQINT8`
- `MQUINT8`
- `MQINT16`
- `MQUINT16`
- `MQINT32`
- `MQUINT32`
- `MQINT64`
- `MQUINT64`

`MQCHAR` 字段已转换代码页，但 `MQBYTE`，`MQINT8` 和 `MQUINT8` 保持不变。如果编码不同，那么将相应地转换 `MQSHORT`，`MQLONG`，`MQINT16`，`MQUINT16`，`MQINT32`，`MQUINT32`，`MQINT64`，`MQUINT64`，`MQFLOAT32`，`MQFLOAT64` 和 `MQBOOL`。

- 请勿使用以下类型的数据：

- 双精度值
- 指针
- 位字段

这是因为用于创建转换-退出代码的实用程序未提供用于转换这些数据类型的工具。要克服此问题，您可以编写自己的例程并从出口调用这些例程。

其他需要注意的问题：

- 请勿在输入数据集中使用序号。
- 如果存在要为其提供自己的转换例程的字段，请将其声明为 `MQBYTE`，然后将生成的 `CMQXCFA` 宏替换为您自己的转换代码。

C 示例

```
struct TEST { MQLONG    SERIAL_NUMBER;
              MQCHAR    ID[5];
              MQINT16   VERSION;
              MQBYTE    CODE[4];
              MQLONG    DIMENSIONS[3];
              MQCHAR    NAME[24];
            } ;
```

这对应于其他编程语言中的以下声明:

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

仅在 z/OS 上受支持

```
DCL 1 TEST,
  2 SERIAL_NUMBER FIXED BIN(31),
  2 ID            CHAR(5),
  2 VERSION       FIXED BIN(15),
  2 CODE          CHAR(4), /* not to be converted */
  2 DIMENSIONS(3) FIXED BIN(31),
  2 NAME         CHAR(24);
```

MQ_PUBLISH_EXIT - 发布出口

MQ_PUBLISH_EXIT 调用可以检查和更改传递给订户的消息。

用途

使用发布出口来检查和更改传递给订户的消息:

- 对发布到每个订户的消息内容进行检查
- 对发布到每个订户的消息内容进行修改
- 更改要将消息放入的队列
- 停止向订户传送消息

语法

MQ_PUBLISH_EXIT(*ExitParms*, *PubContext*, *SubContext*)

参数

ExitParms (MQPSXP) - Input/Output

ExitParms 包含有关出口调用的信息。

PubContext (MQPBC) - Input

PubContext 包含有关出版物发布者的上下文信息。

SubContext (MQSBC) - Input/Output

SubContext 包含有关接收发布的订户的上下文信息。

MQPSXP-发布出口数据结构

MQPSXP 结构描述传递到发布出口并从发布出口返回的信息。

第 909 页的表 589 汇总了结构中的字段:

| 字段 | 描述 |
|----------------------|------------------|
| <i>StrucID</i> | 结构标识 |
| <i>Version</i> | 结构版本号 |
| <i>ExitId</i> | 正在调用的出口的类型 |
| <i>ExitReason</i> | 调用出口的原因 |
| <i>ExitResponse</i> | 来自出口的响应 |
| <i>ExitResponse2</i> | 来自出口的辅助响应 |
| <i>Feedback</i> | 反馈代码 |
| <i>ExitUserArea</i> | 出口用户区域 |
| <i>ExitData</i> | 出口数据 |
| <i>QMgrName</i> | 本地队列管理器的名称 |
| <i>Hconn</i> | 连接句柄 |
| <i>MsgDescPtr</i> | 消息描述符 (MQMD) 的地址 |
| <i>MsgHandle</i> | 消息属性的句柄 (MQHMSG) |
| <i>MsgInPtr</i> | 输入消息的地址 |
| <i>MsgInLength</i> | 输入消息的长度 |
| <i>MsgOutPtr</i> | 输出消息的地址 |
| <i>MsgOutLength</i> | 输出消息的长度 |
| <i>pEntryPoints</i> | MQIEP 结构的地址 |

字段

StrucID (MQCHAR4)

StrucID 是结构标识。值如下所示:

MQPSXP_STRUCID

MQPSXP_STRUCID 是发布出口参数结构的标识。对于 C 编程语言, 还定义了常量 MQPSXP_STRUC_ID_ARRAY; 它具有与 MQPSXP_STRUC_ID 相同的值, 但它是字符数组而不是字符串。

StrucID 是出口的输入字段。

Version (MQLONG)

Version 是结构版本号。值如下所示:

MQPSXP_VERSION_1

MQPSXP_VERSION_1 是 V 1 发布出口参数结构。常量 MQPSXP_CURRENT_VERSION 也使用相同的值进行定义。

Version 是出口的输入字段。

ExitId (MQLONG)

ExitId 是正在调用的出口的类型。值如下所示:

MQXT_PUBLISH_EXIT

发布出口。

ExitId 是出口的输入字段。

ExitReason (MQLONG)

ExitReason 是调用出口的原因。可能的值为:

MQXR_INIT

将调用此连接的出口以进行初始化。出口可能获取并初始化它需要的资源;例如,主存储器。

MQXR_TERM

将调用此连接的出口,因为该出口即将停止。出口必须释放自初始化以来获取的任何资源;例如,主存储器。

MQXR_PUBLICATION

在将发布内容放入订户的消息队列之前,队列管理器将调用该出口。出口可以更改消息,不将消息放入队列或停止发布。

ExitReason 是出口的输入字段。

ExitResponse (MQLONG)

在出口中设置 *ExitResponse* 以指定处理必须如何继续。*ExitResponse* 是下列其中一个值:

MQXCC_OK

设置 MQXCC_OK 以继续正常处理。设置 MQXCC_OK 以响应 *ExitReason* 的任何值。

如果 *ExitReason* 具有值 MQXR_PUBLICATION,那么 MQSBC 结构的 *DestinationQName* 和 *DestinationQMgrName* 字段将标识将消息发送到的目标。

MQXCC_FAILED

设置 MQXCC_FAILED 以停止发布操作。完成代码 MQCC_FAILED 和原因码 2557 (09FD) (RC2557):MQRC_PUBLISH_EXIT_ERROR 设置为从出口返回。

MQXCC_SUPPRESS_FUNCTION

设置 MQXCC_SUPPRESS_FUNCTION 以停止消息的正常处理。仅当 *ExitReason* 具有值 MQXR_PUBLICATION 时,才设置 MQXCC_SUPPRESS_FUNCTION。

队列管理器根据消息描述符的 *Report* 字段中的 MQRO_DISCARD_MSG 选项继续处理该消息。

- 如果指定了 MQRO_DISCARD_MSG 选项,那么不会将消息传递到订户。
- 如果未指定 MQRO_DISCARD_MSG 选项,那么会将消息放在死信队列上。如果没有死信队列,或者消息无法成功放在死信队列上,那么发布不会传递给订户。将发布内容传递到其他订户取决于 PMSGDLV 和 NPMSGDLV 主题对象属性的值。有关这些属性的说明,请参阅 [DEFINE TOPIC](#) 命令的参数描述。

ExitResponse 是输出的输出字段。

ExitResponse2 (MQLONG)

ExitResponse2 保留供将来使用。

Feedback (MQLONG)

Feedback 是在出口返回 *ExitResponse* 中的 MQXCC_SUPPRESS_FUNCTION 时要使用的反馈代码。

在输入到出口时, *Feedback* 始终具有值 MQFB_NONE。如果出口返回 MQXCC_SUPPRESS_FUNCTION,请将 *Feedback* 设置为当队列管理器将其放入死信队列时要用于消息

的值。从出口返回时，如果 *Feedback* 具有原始值 MQFB_NONE，那么队列管理器会将 *Feedback* 设置为 MQFB_STOPPED_BY_PUBSUB_EXIT。

Feedback 是出口的输入/输出字段。

ExitUserArea (MQBYTE16)

ExitUserArea 是可供出口使用的字段。每个连接都有单独的 *ExitUserArea*。*ExitUserArea* 的长度由 MQ_EXIT_USER_AREA_LENGTH 给出。

ExitReason 字段在第一次调用出口时具有值 MQXR_INIT。在第一次调用连接出口时，会将 *ExitUserArea* 初始化为 MQXUA_NONE。在出口的调用中保留对 *ExitUserArea* 的后续更改。

ExitUserArea 是出口的输入/输出字段。

ExitData (MQCHAR32)

ExitData 是由队列管理器初始化文件中节的 *PublishExitData* 参数定义的固定出口数据。用空白填充数据到字段的完整长度。如果在初始化文件中未定义固定出口数据，那么 *ExitData* 为空白。

ExitData 的长度由 MQ_EXIT_DATA_LENGTH 给出。

ExitData 是出口的输入字段。

QMgrName (MQCHAR48)

QMgrName 是本地队列管理器的名称。该名称将用空格填充到字段的完整长度。此字段的长度由 MQ_Q_MGR_NAME_LENGTH 给出。

QMgrName 是出口的输入字段。

Hconn (MQHCONN)

Hconn 是表示与队列管理器的连接的句柄。仅将 *Hconn* 用作 MQSETMP, MQINQMMP 或 MQDLTMP 消息属性函数调用的参数，以使用消息属性。

Hconn 是出口的输入字段。

MsgDescPtr (PMQMD)

MsgDescPtr 是正在处理的消息的消息描述符 (MQMD) 的地址，并且是从 MQPUT 调用返回的 MQMD 的副本。出口可以更改消息描述符的内容。必须小心地完成对消息描述符内容的任何更改。特别是，在 MQSBC 结构的 *SubType* 字段值为 MQSUBTYPE_PROXY 的情况下，不得更改消息描述符中的 *CorrelId* 字段。

如果 *ExitReason* 为 MQXR_INIT 或 MQXR_TERM，那么不会将任何消息描述符传递到出口；在这些情况下，*MsgDescPtr* 是空指针。

MsgDescPtr 是出口的输入字段。

MsgHandle (MQHMSG)

MsgHandle 是消息属性的句柄。仅将 *MsgHandle* 与 MQSETMP, MQINQMMP 或 MQDLTMP 消息属性函数调用配合使用以使用消息属性。

MsgHandle 是出口的输入字段。

MsgInPtr (PMQVOID)

MsgInPtr 是输入消息数据的地址。*MsgInPtr* 寻址的缓冲区内容可由出口修改；请参阅 [MsgOutPtr](#)。

MsgInPtr 是出口的输入字段。

MsgInLength (MQLONG)

MsgInLength 是传递到出口的消息数据的长度 (以字节计)。数据的地址由 *MsgInPtr* 提供。

MsgInLength 是出口的输入字段。

MsgOutPtr (PMQVOID)

MsgOutPtr 是包含从出口返回的消息数据的缓冲区的地址。进入出口时，*MsgOutPtr* 为空。从出口返回时，如果值仍然为空，那么队列管理器将发送由 *MsgInPtr* 指定的消息，其长度由 *MsgInLength* 指定。

如果出口修改消息数据，请使用下列其中一个过程：

- 如果数据长度未更改，那么可以在 *MsgInPtr* 寻址的缓冲区中修改数据。在这种情况下，请勿更改 *MsgOutPtr* 和 *MsgOutLength*。
- 如果修改的数据比原始数据短，那么可以在 *MsgInPtr* 寻址的缓冲区中修改数据。在这种情况下，必须将 *MsgOutPtr* 设置为输入消息缓冲区的地址，并将 *MsgOutLength* 设置为消息数据的新长度。
- 如果修改后的数据比原始数据长，或者可能比原始数据长，那么出口必须获取新的消息缓冲区。将修改后的数据复制到其中。将 *MsgOutPtr* 设置为新缓冲区的地址，并将 *MsgOutLength* 设置为新消息数据的长度。下次调用出口时，该出口负责释放 *MsgOutPtr* 寻址的缓冲区。

注：*MsgOutPtr* 始终是出口输入的空指针，而不是先前获取的消息缓冲区的地址。要释放先前获取的缓冲区，出口必须保存其地址和长度。将信息保存在 *ExitUserArea* 中，或保存在将其地址保存在 *ExitUserArea* 中的控制块中。

MsgOutPtr 是出口的输入/输出字段。

MsgOutLength (MQLONG)

MsgOutLength 是出口返回的消息数据的长度(以字节计)。在出口的输入上，此字段始终为零。从出口返回时，如果 *MsgOutPtr* 为空，那么将忽略此字段。有关修改消息数据的信息，请参阅 [MsgOutPtr](#)。

MsgOutLength 是出口的输入/输出字段。

pEntryPoints (PMQIEP)

pEntryPoints 是 MQIEP 结构的地址，可通过该结构进行 MQI 和 DCI 调用。

C 语言声明-MQPSXP

```
typedef struct tagMQPSXP {
    MQCHAR4      StructId;          /* Structure identifier */
    MQLONG       Version;           /* Structure version number */
    MQLONG       ExitId;            /* Type of exit */
    MQLONG       ExitReason;        /* Reason for invoking exit */
    MQLONG       ExitResponse;      /* Response from exit */
    MQLONG       ExitResponse2;     /* Reserved */
    MQLONG       Feedback;          /* Feedback code */
    MQBYTE16     ExitUserArea;      /* Exit user area */
    MQCHAR32     ExitData;          /* Exit data */
    MQCHAR48     QMgrName;          /* Name of local queue manager */
    MQHCONN      Hconn;             /* Connection handle */
    MQHMSG       MsgHandle;         /* Handle to message properties */
    PMQMD        MsgDescPtr;        /* Address of message descriptor */
    PMQVOID      MsgInPtr;          /* Address of input message data */
    MQLONG       MsgInLength;       /* Length of input message data */
    PMQVOID      MsgOutPtr;         /* Address of output message data */
    MQLONG       MsgOutLength;      /* Length of output message data */
    /* Ver:1 */
    PMQIEP       pEntryPoints;      /* Address of the MQIEP structure */
    /* Ver:2 */
} MQPSXP;
```

MQPBC-发布上下文数据结构

MQPBC 结构包含传递到发布出口的与发布的发布程序相关的上下文信息。

第 912 页的表 590 汇总了结构中的字段：

| 表 590: MQPBC 中的字段 | |
|-----------------------|------------------|
| 字段 | 描述 |
| <i>StructID</i> | 结构标识 |
| <i>Version</i> | 结构版本号 |
| <i>PubTopicString</i> | 发布主题字符串 |
| <i>MsgDescPtr</i> | 消息描述符 (MQMD) 的地址 |

字段

StrucID (MQCHAR4)

StrucID 是结构标识。值如下所示:

MQPBC_STRUCID

MQPBC_STRUCID 是发布上下文结构的标识。对于 C 编程语言, 还定义了常量 MQPBC_STRUC_ID_ARRAY; 它具有与 MQPBC_STRUC_ID 相同的值, 但它是字符数组而不是字符串。

StrucID 是出口的输入字段。

Version (MQLONG)

Version 是结构版本号。值如下所示:

MQPBC_VERSION_1

MQPBC_VERSION_1 是 V 1 发布出口参数结构。

MQPBC_VERSION_2

MQPBC_VERSION_2 是 V 2 发布出口参数结构。常量 MQPBC_CURRENT_VERSION 也使用相同的值进行定义。

Version 是出口的输入字段。

PubTopicString (MQCHARV)

PubTopicString 是要发布到的主题字符串。

PubTopicString 是出口的输入字段。

MsgDescPtr (PMQMD)

MsgDescPtr 是正在处理的的消息的消息描述符 (MQMD) 副本的地址。

MsgDescPtr 是出口的输入字段。

C 语言声明-MQPBC

```
typedef struct tagMQPBC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHARV    PubTopicString;   /* Publish topic string */
    PMQMD      MsgDescPtr;      /* Address of message descriptor */
} MQPBC;
```

MQSBC-预订上下文数据结构

MQSBC 结构包含与正在接收发布的订户相关的上下文信息, 这些信息将传递到发布出口。

第 913 页的表 591 汇总了结构中的字段:

| 字段 | 描述 |
|----------------------------|------------|
| <u>StrucID</u> | 结构标识 |
| <u>Version</u> | 结构版本号 |
| <u>DestinationQMgrName</u> | 目标队列管理器的名称 |
| <u>DestinationQName</u> | 目标队列的名称 |
| <u>SubType</u> | 预订的类型 |
| <u>SubOptions</u> | 预订选项 |
| <u>ObjectName</u> | 对象名 |
| <u>ObjectString</u> | 对象字符串 |

| 表 591: 字段(F) MQSBC (继续) | |
|-------------------------|----------|
| 字段 | 描述 |
| <u>SubTopicString</u> | 预订主题字符串 |
| <u>SubName</u> | 预订名称 |
| <u>SubId</u> | 预订标识 |
| <u>SelectionString</u> | 选择字符串的地址 |
| <u>SubLevel</u> | 预订级别 |
| <u>PSProperties</u> | 发布/预订属性 |

字段

StrucID (MQCHAR4)

结构标识。值如下所示:

MQSBC_STRUCID

MQSBC_STRUCID 是发布出口参数结构的标识。对于 C 编程语言, 还定义了常量 MQSBC_STRUC_ID_ARRAY; MQSBC_STRUC_ID_ARRAY 具有与 MQSBC_STRUC_ID 相同的值, 但是字符数组而不是字符串。

StrucID 是出口的输入字段。

Version (MQLONG)

结构版本号。值如下所示:

MQSBC_VERSION_1

V 1 发布出口参数结构。常量 MQSBC_CURRENT_VERSION 也使用相同的值进行定义。

Version 是出口的输入字段。

DestinationQMgrName (MQCHAR48)

DestinationQMgrName 是要向其发送消息的队列管理器的名称。该名称将用空格填充到字段的完整长度。该名称可由出口更改。此字段的长度由 MQ_Q_MGR_NAME_LENGTH 给出。

DestinationQMgrName 是出口的输入/输出字段; 请参阅 [注释](#)。

DestinationQName (MQCHAR48)

DestinationQName 是要将消息发送到的队列的名称。该名称将用空格填充到字段的完整长度。该名称可由出口更改。此字段的长度由 MQ_Q_NAME_LENGTH 给出。

DestinationQName 是出口的输入/输出字段; 请参阅 [注释](#)。

SubType (MQLONG)

SubType 指示如何创建预订。有效值为 MQSUBTYPE_API, MQSUBTYPE_ADMIN 和 MQSUBTYPE_PROXY; 请参阅 [查询预订状态 \(响应\)](#)。

SubType 是出口的输入字段。

SubOptions (MQLONG)

SubOptions 是预订选项; 请参阅 [第 494 页的『选项 \(MQLONG\)』](#) 以获取此字段可采用的值的描述。

SubOptions 是出口的输入字段。

ObjectName (MQCHAR48)

ObjectName 是在本地队列管理器上定义的主题对象的名称。此字段的长度由 MQ_TOPIC_NAME_LENGTH 给出。对象名是队列管理器与主题字符串关联的管理主题对象的名称。即使订户在预订过程中提供了主题对象, *ObjectName* 也可能是另一主题对象。主题对象与预订的关联取决于 *SubTopicString* 的完整解析。

ObjectName 是出口的输入字段。

ObjectString (MQCHARV)

ObjectString 是预订的发布的完整主题字符串。将解析原始预订字符串中的任何通配符。它与第 493 页的『ObjectString (MQCHARV)』中描述的 MQSD 预订 *ObjectString* 字段不同，后者可能包含通配符，并且不包含订户提供的任何对象名。

ObjectString 是出口的输入字段。

SubTopicString (MQCHARV)

SubTopicString 是订户提供的完整主题字符串。*SubTopicString* 是主题对象中定义的主题字符串与主题字符串的组合。订户必须提供主题对象和/或主题字符串。如果订户提供主题字符串，那么可能包含通配符。

SubTopicString 是出口的输入字段。

SubName (MQCHARV)

SubName 是订户提供的预订名称，或者是生成的名称。

SubName 是出口的输入字段。

SubId (MQBYTE 24)

SubId 是唯一的内部预订标识。

SubId 是出口的输入字段。

SelectionString (MQCHARV)

SelectionString 是预订来自主题的消息时使用的选择标准；请参阅[选择器](#)。

SelectionString 是出口的输入字段。

SubLevel (MQLONG)

SubLevel 是与预订关联的拦截级别；请参阅第 503 页的『SubLevel (MQLONG)』以获取更多详细信息。

SubLevel 是出口的输入字段。

PSPProperties (MQLONG)

PSPProperties 是发布/预订属性。它们指定如何将与发布/预订相关的消息属性添加到发送到此预订的消息。可能的值为 MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2 和 MQPSPROP_MSGPROP。请参阅[可选参数 \(更改, 复制和创建预订\)](#) 以获取这些值的描述。

PSPProperties 是出口的输入字段。

注: 仅在将 *DestinationQMgrName* 和 *DestinationQName* 的原始值传递到发布出口之前，才会对其执行授权检查。出口通过更改 *DestinationQMgrName* 或 *DestinationQName* 来更改目标队列时，不会执行新的授权检查。

C 语言声明-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       PSPProperties;    /* Publish/subscribe properties */
} MQSBC;
```

通道出口调用和数据结构

此主题集合提供有关编写通道出口程序时可以使用的特殊 WebSphere MQ 调用和数据结构的参考信息。

此信息是产品敏感的编程接口信息。您可以使用以下编程语言编写 WebSphere MQ 用户出口:

| 平台 | 编程语言 |
|------------------------|--|
| WebSphere MQ for z/OS | 汇编程序和 C (必须符合系统出口的 C 系统编程环境, 如 <i>z/OS C/C++ Programming Guide</i> 中所述。) |
| WebSphere MQ for IBM i | ILE C, ILE COBOL 和 ILE RPG |
| 所有其他 WebSphere MQ 平台 | C |

您还可以在 Java 中编写用户出口, 以便仅与 Java 和 JMS 应用程序配合使用。有关使用 WebSphere MQ classes for Java 创建和使用通道出口的更多信息, 请参阅 [在 WebSphere MQ classes for Java 中使用通道出口](#), 对于 WebSphere MQ classes for JMS, 请参阅 [将通道出口与 WebSphere MQ classes for JMS 配合使用](#)。

不能使用 TAL 或 Visual Basic 编写 WebSphere MQ 用户出口。但是, 在 Visual Basic 中提供了 MQCD 结构的声明, 以用于来自 WebSphere MQ MQI 客户机程序的 MQCONN 调用。

在以下描述中的许多情况下, 参数是大小不固定的数组或字符串。对于这些参数, 将使用小写 “n” 来表示数字常量。当对该参数的声明进行编码时, 必须将 “n” 替换为所需的数字值。有关这些描述中使用的约定的更多信息, 请参阅 [第 214 页的『基本数据类型』](#)。

数据定义文件

WebSphere MQ 为每种受支持的编程语言提供了数据定义文件。有关这些文件的详细信息, 请参阅 [复制, 包含和模块文件](#)。

MQ_CHANNEL_EXIT-通道出口

MQ_CHANNEL_EXIT 调用描述传递到消息通道代理程序调用的每个通道出口的参数。

队列管理器未提供名为 MQ_CHANNEL_EXIT 的入口点; 由于通道定义 MQCD 中提供了通道出口的名称, 因此名称 MQ_CHANNEL_EXIT 没有特殊意义。

有五种类型的通道出口:

- 通道安全出口
- 通道消息出口
- 通道发送出口
- 通道接收出口
- 通道消息-重试出口

对于每种类型的出口, 这些参数都是相似的, 此处给出的描述适用于所有这些出口, 但有特别说明的除外。

语法

MQ_CHANNEL_EXIT (ChannelExitParms, ChannelDefinition, DataLength, AgentBufferLength, AgentBuffer, ExitBufferLength, ExitBufferAddr)

参数

MQ_CHANNEL_EXIT 调用具有以下参数。

ChannelExit 参数 (MQCXP)-输入/输出

通道出口参数块。

此结构包含与出口调用相关的其他信息。出口设置此结构中的信息以指示 MCA 的继续方式。

ChannelDefinition (MQCD)-输入/输出

通道定义。

此结构包含管理员为控制通道行为而设置的参数。

DataLength (MQLONG)-输入/输出

数据长度。

数据取决于出口类型:

- 对于通道安全出口, 当调用该出口时, 如果 *ExitReason* 是 *MQXR_SEC_MSG*, 那么此参数在 *AgentBuffer* 字段中包含任何安全消息的长度。如果没有消息, 那么为零。如果将 *ExitResponse* 设置为 *MQXCC_SEND_SEC_MSG* 或 *MQXCC_SEND_AND_REQUEST_SEC_MSG*, 那么出口必须将此字段设置为要发送给其合作伙伴的任何安全消息的长度。消息数据位于 *AgentBuffer* 或 *ExitBufferAddr* 中。

安全消息的内容由安全出口自行负责。

- 对于通道消息出口, 当调用该出口时, 此参数包含消息的长度 (包括传输队列头)。出口必须将此字段设置为要继续的 *AgentBuffer* 或 *ExitBufferAddr* 中的消息长度。这必须大于或等于传输队列头 (*MQXQH*) 的长度。
- 对于通道发送或通道接收出口, 当调用该出口时, 此参数包含传输的长度。出口必须将此字段设置为要继续的 *AgentBuffer* 或 *ExitBufferAddr* 中的传输长度。

如果安全出口发送消息, 并且在通道的另一端没有安全出口, 或者另一端设置了 *MQXCC_OK* 的 *ExitResponse*, 那么将使用 *MQXR_SEC_MSG* 和空响应 (*DataLength*= 0) 重新调用启动出口。

AgentBuffer 长度 (MQLONG)-输入

代理程序缓冲区的长度。

此参数在调用时可以大于 *DataLength*。

对于通道消息, 发送和接收出口, 该出口可以使用调用时的任何未使用空间来扩展现有数据。如果已完成此操作, 那么 *DataLength* 参数必须由出口进行相应设置。

在 C 编程语言中, 此参数按地址传递。

AgentBuffer (MQBYTE × AgentBuffer 长度)-输入/输出

代理程序缓冲区。

此参数的内容取决于出口类型:

- 对于通道安全出口, 在调用该出口时, 如果 *ExitReason* 是 *MQXR_SEC_MSG*, 那么该出口包含安全消息。要将安全消息发送回, 出口可以使用此缓冲区或其自己的缓冲区 (*ExitBufferAddr*)。
- 对于通道消息出口, 在调用该出口时, 此参数包含:

- 传输队列头 (*MQXQH*), 它包含消息描述符 (其本身包含消息的上下文信息), 紧跟其后
- 消息数据

如果消息要继续, 那么出口可以执行下列其中一项操作:

- 使缓冲区的内容保持不变
- 修改现有内容 (在 *DataLength* 中返回数据的新长度; 此长度不得大于 *AgentBufferLength*)
- 将内容复制到 *ExitBufferAddr*, 进行任何必需的更改

不会检查出口对传输队列头所作的任何更改; 但是, 错误修改可能意味着无法将消息放入目标。

- 对于通道发送或接收出口, 在调用该出口时, 此出口包含传输数据。出口可以执行下列其中一项操作:
 - 使缓冲区的内容保持不变
 - 修改现有内容 (在 *DataLength* 中返回数据的新长度; 此长度不得大于 *AgentBufferLength*)
 - 将内容复制到 *ExitBufferAddr*, 进行任何必需的更改

出口不得更改数据的前 8 个字节。

ExitBuffer 长度 (MQLONG)-输入/输出

出口缓冲区的长度。

在第一次调用出口时，此参数设置为零。此后，每次调用时，出口传递回的任何值都将在下次调用时显示到出口。MCA 不使用该值。

注：以不支持指针数据类型的编程语言编写的出口不得使用此参数。

ExitBufferAddr (MQPTR)-输入/输出

出口缓冲区的地址。

此参数是指向出口所管理的存储器的缓冲区地址的指针，在该地址中，如果代理的缓冲区足够大或可能不够大，或者如果出口比较方便，那么它可以选择向代理返回消息或传输数据（取决于出口的类型）。

在第一次调用出口时，传递到该出口的地址为空。此后，在每次调用时，出口传递回的任何地址都将在下次调用时提供给出口。

注：以不支持指针数据类型的编程语言编写的出口不得使用此参数。

C 调用

```
exitname (&ChannelExitParms, &ChannelDefinition,  
          &DataLength, &AgentBufferLength, AgentBuffer,  
          &ExitBufferLength, &ExitBufferAddr);
```

传递到出口的参数声明如下：

```
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 */
```

COBOL 调用

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,  
                     DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,  
                     EXITBUFFERLENGTH, EXITBUFFERADDR.
```

传递到出口的参数声明如下：

```
** 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 调用 (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..  
C          CALLP      exitname(MQCXP : MQCD : DATLEN :  
C                               ABUFL : ABUF : EBUFL :  
C                               EBUF)
```

调用的原型定义为:

```
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 汇编程序调用

```
CALL EXITNAME, (CHANNELEXITPARMS, CHANNELDEFINITION, DATALENGTH, X
                AGENTBUFFERLENGTH, AGENTBUFFER, EXITBUFFERLENGTH, X
                EXITBUFFERADDR)
```

传递到出口的参数声明如下:

| | | | |
|-------------------|---------|-------|------------------------------|
| 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 |

使用说明

1. 通道出口执行的函数由出口的提供程序定义。但是，出口必须符合此处定义的规则以及关联的控制块 MQCXP 中的规则。
2. 传递到通道出口的 *ChannelDefinition* 参数可能是多个版本之一。请参阅 MQCD 结构中的 *Version* 字段以获取更多信息。
3. 如果通道出口接收到将 *Version* 字段设置为大于 MQCD_VERSION_1 的值的 MQCD 结构，那么该出口必须使用 MQCD 中的 *ConnectionName* 字段 (优先于 *ShortConnectionName* 字段)。
4. 通常，允许通道出口更改消息数据的长度。这可能是由于出口将数据添加到消息，或者从消息中除去数据，或者压缩或加密消息而产生的。但是，如果消息是仅包含部分逻辑消息的段，那么适用特殊限制。特别是，不得由于补充发送和接收出口的操作而使消息长度发生净变化。

例如，允许发送出口通过压缩来缩短消息，但补充接收出口必须通过解压缩来恢复消息的原始长度，从而使消息的长度没有净变化。

产生此限制的原因是更改段的长度将导致消息中的后续段的偏移量不正确，这将阻止队列管理器识别这些段构成完整逻辑消息的能力。

MQ_CHANNEL_AUTO_DEF_EXIT-通道自动定义出口

MQ_CHANNEL_AUTO_DEF_EXIT 调用描述传递到消息通道代理程序调用的通道自动定义出口的参数。

队列管理器未提供名为 MQ_CHANNEL_AUTO_DEF_EXIT 的入口点; 由于在队列管理器中提供了自动定义出口的名称，因此名称 MQ_CHANNEL_AUTO_DEF_EXIT 没有特殊意义。

语法

MQ_CHANNEL_AUTO_DEF_EXIT (*ChannelExitParms*, *ChannelDefinition*)

参数

MQ_CHANNEL_AUTO_DEF_EXIT 调用具有以下参数。

ChannelExit 参数 (MQCXP)-输入/输出

通道出口参数块。

此结构包含与出口调用相关的其他信息。出口设置此结构中的信息以指示 MCA 的继续方式。

ChannelDefinition (MQCD)-输入/输出

通道定义。

此结构包含管理员设置的参数，用于控制自动创建的通道的行为。出口设置此结构中的信息以修改管理员设置的缺省行为。

出口不得改变列出的 MQCD 字段:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

如果更改了其他字段，那么出口设置的值必须有效。如果该值无效，那么会将错误消息写入错误日志文件或显示在控制台上 (视环境而定)。

C 调用

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

传递到出口的参数声明如下:

```
MQCXP  ChannelExitParms; /* Channel exit parameter block */
MQCD   ChannelDefinition; /* Channel definition */
```

COBOL 调用

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

传递到出口的参数声明如下:

```
** Channel exit parameter block
01 CHANNELEXITPARMS.
   COPY CMQXPV.
** Channel definition
01 CHANNELDEFINITION.
   COPY CMQCDV.
```

RPG 调用 (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..
C          CALLP      exitname(MQCXP : MQCD)
```

调用的原型定义为:


```

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 汇编程序调用

```
CALL EXITNAME, (CHANNELEXITPARMS, CHANNELDEFINITION)
```

传递到出口的参数声明如下:

```

CHANNELEXITPARMS  CMQCXPA  , Channel exit parameter block
CHANNELDEFINITION CMQCDA   , Channel definition

```

使用说明

1. 通道出口执行的函数由出口的提供程序定义。但是，出口必须符合此处定义的规则以及关联的控制块 MQCXP 中的规则。
2. 传递到通道自动定义出口的 *ChannelExitParms* 参数是 MQCXP 结构。传递的 MQCXP 版本取决于运行出口的环境; 请参阅第 958 页的『MQCXP-通道出口参数』中 *Version* 字段的描述以获取详细信息。
3. 传递到通道自动定义出口的 *ChannelDefinition* 参数是 MQCD 结构。传递的 MQCD 版本取决于运行出口的环境; 请参阅第 922 页的『MQCD-通道定义』中 *Version* 字段的描述以获取详细信息。

MQXWAIT-等待退出

MQXWAIT 调用等待事件发生。它只能从 z/OS 上的通道出口使用。

使用 MQXWAIT 有助于避免在通道出口执行导致等待的操作时可能发生的性能问题。正在等待的事件 MQXWAIT 由 MVS ECB (事件控制块) 发出信号。在 MQXWD 控制块描述中描述了 ECB。

语法

MQXWAIT (Hconn, WaitDesc, CompCode, Reason)

参数

MQXWAIT 调用具有以下参数。

Hconn (MQHCONN)-输入

连接句柄。

此句柄表示与队列管理器的连接。Hconn 的值由在出口的相同或较早调用中发出的先前 MQCONN 调用返回。

WaitDesc (MQXWD)-输入/输出

等待描述符。

此参数描述要等待的事件。请参阅第 971 页的『MQXWD-出口等待描述符』以获取此结构中的字段的详细信息。

CompCode (MQLONG)-输出

完成代码。

它是下列其中一个代码:

MQCC_OK
成功完成。

MQCC_FAILED
调用失败。

原因 (MQLONG)-输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE
(0, X'000') 没有要报告的原因。

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') 适配器不可用。

MQRC_OPTIONS_ERROR
(2046, X'7FE') 选项无效或不一致。

MQRC_XWAIT_CANCEL
(2107, X'83B') MQXWAIT 调用已取消。

MQRC_XWAIT_ERROR
(2108, X'83C') 调用 MQXWAIT 调用无效。

C 调用

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

按如下所示声明参数:

```
MQHCONN  Hconn;      /* Connection handle */  
MQXWD    WaitDesc;  /* Wait descriptor */  
MQLONG   CompCode;  /* Completion code */  
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

System/390 汇编程序调用

```
CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

按如下所示声明参数:

```
HCONN      DS      F  Connection handle  
WAITDESC   CMQXWDA ,  Wait descriptor  
COMPCODE   DS      F  Completion code  
REASON     DS      F  Reason code qualifying COMPCODE
```

MQCD-通道定义

MQCD 结构包含用于控制通道执行的参数。它将传递到从消息通道代理程序 (MCA) 调用的每个通道出口。

有关通道出口的更多信息, 请参阅第 916 页的『MQ_CHANNEL_EXIT-通道出口』。本主题中的描述同时与消息通道和 MQI 通道相关。

出口名称字段

调用出口时, 来自 *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExit* 和 *MsgRetryExit* 的相关字段包含当前正在调用的出口的名称。这些字段中名称的含义取决于运行 MCA 的环境。除非另有说明, 否则该名称在字段中左对齐, 没有嵌入的空格; 该名称在字段的长度上用空格填充。在下面的描述中, 方括号 ([]) 表示可选信息:

UNIX 系统

出口名称是动态可装入模块或库的名称，后缀为驻留在该库中的函数的名称。函数名必须括在括号内。可以选择以目录路径作为库名的前缀：

```
[path]library(function)
```

名称限制为最多 128 个字符。

z/OS

出口名称是对 LINK 或 LOAD 宏的 EP 参数规范有效的装入模块的名称。名称限制为最多 8 个字符。

Windows

出口名称是动态链接库的名称，后缀为驻留在该库中的函数的名称。函数名必须括在括号内。可以选择以目录路径和磁带机作为库名的前缀：

```
[d:][path]library(function)
```

名称限制为最多 128 个字符。

IBM i

出口名称是后跟 10 字节库名的 10 字节程序名。如果名称长度小于 10 个字节，那么将用空格填充每个名称以使其为 10 个字节。库名可以是 *LIBL，但调用通道自动定义出口时除外，在这种情况下需要标准名称。

更改通道出口中的 MQCD 字段

通道出口可以更改 MQCD 中的字段。更改后的值将保留在 MQCD 中，并传递到出口链中的任何剩余出口以及共享通道实例的任何对话。在通道的持续生存期内，MCA 还会使用已更改的 MQCD 进行其正常处理。

出口不得改变以下 MQCD 字段：

- ChannelName
- ChannelType
- StrucLength
- 版本

相关参考

[第 923 页的『字段』](#)

本主题列出 MQCD 结构中的所有字段并描述每个字段。

[第 945 页的『C 声明』](#)

此声明是 MQCD 结构的 C 声明。

[第 947 页的『COBOL 声明』](#)

此声明是 MQCD 结构的 COBOL 声明。

[第 949 页的『RPG 声明 \(ILE\)』](#)

此声明是 MQCD 结构的 RPG 声明。

[第 952 页的『System/390 汇编程序声明』](#)

此声明是 MQCD 结构的 System/390 汇编程序声明。

[第 954 页的『Visual Basic 声明』](#)

此声明是 MQCD 结构的 Visual Basic 声明。

[第 955 页的『更改通道出口中的 MQCD 字段』](#)

通道出口可以更改 MQCD 中的字段。但是，通常不会对这些更改执行操作，除非在列出的情况下。

字段

本主题列出 MQCD 结构中的所有字段并描述每个字段。

BatchHeartbeat (MQLONG)

此字段指定用于触发通道的批处理脉动信号的时间间隔。

批处理脉动信号允许发送方通道在不确定之前确定远程通道实例是否仍处于活动状态。如果发送方通道在指定的时间间隔内未与远程通道实例通信，那么将发生批处理脉动信号。

该值在范围 0 到 999 999 999 之间；单位为毫秒。值为零表示未启用批处理脉动信号。

此字段仅与 *ChannelType* 为 MQCHT_SENDER，MQCHT_SERVER，MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_7，那么该字段不存在。

BatchInterval (MQLONG)

此字段指定通道保持批处理打开的大致时间 (以毫秒为单位) (如果在当前批处理中传输的消息数少于 *BatchSize*)。

如果 *BatchInterval* 大于零，那么批处理将由首先发生的以下事件中的任何事件终止：

- 已发送 *BatchSize* 条消息，或者
- 自批处理启动以来已经过 *BatchInterval* 毫秒。

如果 *BatchInterval* 为零，那么将首先发生以下任一事件来终止批处理：

- 已发送 *BatchSize* 条消息，或者
- 传输队列变为空。

BatchInterval 必须在范围 0 到 999 999 999 之间。

此字段仅与 *ChannelType* 为 MQCHT_SENDER，MQCHT_SERVER，MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

这是出口的输入字段。当 *Version* 小于 MQCD_VERSION_4 时，该字段不存在。

BatchSize (MQLONG)

此字段指定在同步通道之前可以通过通道发送的最大消息数。

此字段与 *ChannelType* 为 MQCHT_SVRCONN 或 MQCHT_CLNTCONN 的通道无关。

ChannelMonitoring (MQLONG)

此字段指定通道的当前监视数据收集级别。

此字段与 *ChannelType* 为 MQCHT_CLNTCONN 的通道无关。

它是下列其中一个值：

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_8，那么它不存在。

ChannelName (MQCHAR20)

此字段指定通道定义名称。

在远程机器上必须有同名的通道定义才能进行通信。

名称必须仅使用以下字符：

- 大写字母 A-Z
- 小写字母 a-z
- 数字 0-9
- 句点 (.)
- 正斜杠 (/)

- 下划线 (_)
- 百分号 (%)

用空格垫在右边 不允许前导空白或嵌入空白。

此字段的长度由 MQ_CHANNEL_NAME_LENGTH 指定。

ChannelStatistics (MQLONG)

此字段指定通道的当前统计信息数据收集级别。

此字段与 ChannelType 为 MQCHT_CLNTCONN 的通道无关。

它是下列其中一个值:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

这是出口的输出字段。如果 *Version* 小于 MQCD_VERSION_8, 那么它不存在。

ChannelType (MQLONG)

此字段指定通道的类型。

它是下列其中一个值:

MQCHT_SENDER

发送方。

MQCHT_SERVER

服务器。

MQCHT_RECEIVER

接收器。

MQCHT_REQUESTER

请求者。

MQCHT_CLNTCONN

客户机连接。

MQCHT_SVRCONN

服务器连接 (供客户机使用)。

MQCHT_CLUSSDR

集群发送方。

MQCHT_CLUSRCVR

集群接收方。

ClientChannel 权重 (MQLONG)

此字段指定影响所使用的客户机连接通道定义的权重。

使用 ClientChannel 权重属性, 以便在有多个合适的定义可用时, 可以根据其权重随机选择客户机通道定义。当客户机发出 MQCONN 请求与队列管理器组的连接时, 通过指定以星号开头的队列管理器名称, 并在客户机通道定义表 (CCDT) 中提供多个合适的通道定义时, 将根据权重随机选择要使用的定义, 并首先按字母顺序选择任何适用的 ClientChannel 权重 (0) 定义。

请指定 0 至 99 范围内的值。缺省值是 0。

零值指示不执行负载均衡并且按字母顺序选择适用的定义。要启用负载均衡, 请指定一个在范围 1 至 99 之间的值, 其中 1 是最小的权重, 99 是最大的权重。具有非零权重的两个或多个通道之间的消息分布与这些权重的比率成正比。例如, 选择了三个具有 ClientChannel 权重值 2, 4 和 14 的通道, 大约选择了 10%, 20% 和 70% 的时间。不保证此分布。

此属性仅对客户机连接通道类型有效。

这是出口的输出字段。如果 *Version* 小于 MQCD_VERSION_9, 那么该字段不存在。

ClusterPtr (MQPTR)

此字段指定集群名称列表的地址。

如果 *ClustersDefined* 大于零，那么此地址是集群名称列表的地址。通道属于列出的每个集群。

此字段仅与 *ChannelType* 为 MQCMT_CLUSSDR 或 MQCMT_CLUSRCVR 的通道相关。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_5，那么此字段不存在。

ClustersDefined (MQLONG)

此字段指定通道所属的集群数。

此字段是 *ClusterPtr* 指向的集群名称数。它为零或更大。

此字段仅与 *ChannelType* 为 MQCMT_CLUSSDR 或 MQCMT_CLUSRCVR 的通道相关。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_5，那么此字段不存在。

CLWLChannelPriority (MQLONG)

此字段指定集群工作负载通道优先级。

工作负载管理器选择算法从基于列组选择的目标集中选择具有最高优先级的目标。如果有两个可能的目标队列管理器，那么可以使用此属性将一个队列管理器故障转移到另一个队列管理器上。所有消息都将转至具有最高优先级的队列管理器，直到该操作结束为止，然后这些消息将转至具有下一个最高优先级的队列管理器。

该值在 0 到 9 范围内。缺省值是 0。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_8，那么该字段不存在。

有关更多信息，请参阅 [配置队列管理器集群](#)。

CLWLChannelRank (MQLONG)

此字段指定集群工作负载通道列组。

工作负载管理器选择算法选择具有最高列组的目标。当最终目标是另一集群上的队列管理器时，您可以设置中间网关队列管理器的列组（位于相邻集群的交集处），以便选择算法正确选择距离最终目标更近的目标队列管理器。

该值在 0 到 9 范围内。缺省值是 0。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_8，那么该字段不存在。

有关更多信息，请参阅 [配置队列管理器集群](#)。

CLWLChannelWeight (MQLONG)

此字段指定集群工作负载通道权重。

集群工作负载通道权重。

工作负载管理器选择算法使用通道的 "权重" 属性来偏差目标选项，以便可以将更多消息发送到特定机器。例如，可以给大型 UNIX 服务器上的通道提供比小型桌面 PC 上的另一个通道更大的 "权重"，选择算法选择 UNIX 服务器的频率比 PC 更高。

该值在范围 1 到 99 之间。缺省值为 50。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_8，那么该字段不存在。

有关更多信息，请参阅 [配置队列管理器集群](#)。

ConnectionAffinity (MQLONG)

此字段指定使用同一队列管理器名称多次连接的客户机应用程序是否使用同一客户机通道。

当存在多个适用的通道定义时，请使用此属性。

该值为下列其中之一：

首选 MQAFTY_PREFERRED

读取客户机通道定义表 (CCDT) 的进程中的第一个连接将根据使用任何适用的 CLNTWGHT (0) 定义的权重, 以字母顺序创建适用定义的列表。进程中的每个连接尝试使用该列表中的第一个定义进行连接。如果连接不成功, 那么将使用下一个定义。具有 0 以外的 CLNTWGHT 值的不成功定义将移至列表的末尾。CLNTWGHT(0)定义仍留在列表开头, 并且先对每个连接选中。

具有相同主机名的每个客户机进程总是创建相同的列表。

对于使用 C, C++ 或 .NET 编程框架 (包括完全受管的 .NET) 编写的客户机应用程序, 如果自创建列表以来已修改 CCDT, 那么将更新该列表。

该值为缺省值。

MQAFTY_NONE

进程中读取 CCID 的第一个连接创建适用的定义列表。进程中的所有连接将根据权重及按字母顺序选中的所有适用 CLNTWGHT(0)定义来选择适用定义。

对于使用 C, C++ 或 .NET 编程框架 (包括完全受管的 .NET) 编写的客户机应用程序, 如果自创建列表以来已修改 CCDT, 那么将更新该列表。

此属性仅对客户机连接通道类型有效。

这是出口的输出字段。如果 *Version* 小于 MQCD_VERSION_9, 那么该字段不存在。

ConnectionName (MQCHAR264)

此字段指定通道的连接名称。

对于集群接收方通道 (指定时), CONNAME 与本地队列管理器相关, 而对于其他通道, CONNAME 与目标队列管理器相关。您指定的值取决于要使用的传输协议 (*TransportType*):

- 对于 MQXPT_LU62, 它是伙伴逻辑单元的标准名称。
- 对于 MQXPT_NETBIOS, 它是在远程机器上定义的 NetBIOS 名称。
- 对于 MQXPT_TCP, 它是主机名, 以 IPv4 点分十进制或 IPv6 十六进制格式指定的远程机器的网络地址, 或者是集群接收方通道的本地机器。
- 对于 MQXPT_SPX, 它是包含 4 字节网络地址, 6 字节节点地址和 2 字节套接字号的 SPX 样式地址。

定义通道时, 此字段与 *ChannelType* 为 MQCHT_SVRCONN 或 MQCHT_RECEIVER 的通道无关。但是, 将通道定义传递到出口时, 此字段包含合作伙伴的地址 (无论通道类型如何)。

此字段的长度由 MQ_CONN_NAME_LENGTH 给出。如果 *Version* 小于 MQCD_VERSION_2, 那么此字段不存在。

DataConversion (MQLONG)

此字段指定当接收消息通道代理程序无法执行此转换时, 发送消息通道代理程序是否尝试转换应用程序消息数据。

此字段仅适用于不是逻辑消息段的消息; MCA 从不尝试转换属于段的消息。

此字段仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。它是下列项之一:

MQCDC_SENDER_CONVERSION

按发送方进行转换。

MQCDC_NO_SENDER_CONVERSION

没有按发件人进行转换。

DefReconnect (MQLONG)

DefReconnect 通道属性设置客户机连接通道的缺省重新连接属性值。

缺省自动客户机重新连接选项。您可以配置 IBM WebSphere MQ MQI client 以自动重新连接客户机应用程序。在连接失败后, IBM WebSphere MQ MQI client 会尝试重新连接到队列管理器。它将在无需应用程序客户机发出 MQCONN 或 MQCONNX MQI 调用的情况下尝试重新连接。

重新连接是 MQCONNX 选项。通过使用 DefReconnect 通道属性, 可以将重新连接行为添加到使用 MQCONN 的现有应用程序。您还可以更改使用 MQCONNX 的应用程序的重新连接行为。

您还可以从 `mqclient.ini` 文件设置 `DefRecon` 值，以设置或修改重新连接行为。`mqclient.ini` 文件中的 `DefRecon` 值优先于 `DefReconnect` 通道属性。

Syntax

```
DefReconnect( MQRCN_NO | MQRCN_YES | MQRCN_Q_MGR |
MQRCN_DISABLED)
```

参数

MQRCN_NO

`MQRCN_NO` 是缺省值。

除非被 `MQCONN` 覆盖，否则不会自动重新连接客户机。

MQRCN_YES

除非被 `MQCONN` 覆盖，否则客户机将自动重新连接。

MQRCN_Q_MGR

除非被 `MQCONN` 覆盖，否则客户机将自动重新连接，但仅连接到同一队列管理器。`QMGR` 选项具有与 `MQCNO_RECONNECT_Q_MGR` 相同的效果。

MQRCN_DISABLED

即使客户机程序使用 `MQCONN` `MQI` 调用请求，也会禁用重新连接。

IBM WebSphere MQ Java 类不支持自动客户机重新连接。

| DefReconnect | 应用程序中设置的重新连接选项 | | | |
|----------------|---------------------|---------------------------|----------------------------|------------------------------|
| | MQCNO_RECONNE CT | MQCNO_RECONNE CT_Q_MGR | MQCNO_RECONNE CT_AS_DEF | MQCNO_RECONNE CT_DISABLED |
| MQRCN_NO | YES | QMGR | 否 | 否 |
| MQRCN_YES | YES | QMGR | YES | 否 |
| MQRCN_Q_MGR | YES | QMGR | QMGR | 否 |
| MQRCN_DISABLED | 否 | 否 | 否 | 否 |

相关概念

[自动客户机重新连接](#)

[通道和客户机重新连接](#)

[客户机配置文件的 CHANNELS 节](#)

相关参考

[连接选项](#)

控制 `MQCONN` 操作的选项。

描述 (MQCHAR64)

此字段可用于描述性注释。

该字段的内容对消息通道代理程序没有任何意义。但是，它必须仅包含可显示的字符。它不能包含任何空字符；如有必要，将用空格填充到右边。在 DBCS 安装中，该字段可包含 DBCS 字符 (最大字段长度为 64 字节)。

注: 如果此字段包含不在队列管理器字符集中的字符 (如 `CodedCharSetId` 队列管理器属性所定义)，那么如果将此字段发送到另一个队列管理器，那么这些字符可能转换不正确。

此字段的长度由 `MQ_CHANNEL_DESC_LENGTH` 提供。

DiscInterval (MQLONG)

此字段指定通道在终止通道之前等待消息到达传输队列的最长时间 (以秒计)。

换言之，它指定断开连接时间间隔。

A 值为零将导致 MCA 无限期等待。

对于使用 TCP 协议的服务器连接通道，时间间隔表示以秒为单位指定的客户机不活动断开连接值。如果服务器连接在此持续时间内未接收到来自其伙伴客户机的通信，那么它将终止该连接。服务器连接不活动时间间隔仅在来自客户机的 WebSphere MQ API 调用之间适用，因此在长时间运行的 MQGET 使用等待调用期间，不会断开任何客户机的连接。

此属性不适用于使用 TCP 以外的协议的服务器连接通道。

此字段仅与 *ChannelType* 为 MQCHT_SENDER，MQCHT_SERVER，MQCHT_CLUSSDR，MQCHT_CLUSRCVR 或 MQCHT_SVRCONN 的通道相关。

ExitData 长度 (MQLONG)

此字段指定由 *MsgUserDataPtr*，*SendUserDataPtr* 和 *ReceiveUserDataPtr* 字段寻址的出口用户数据项列表中每个用户数据项的长度 (以字节计)。

此长度不一定与 MQ_EXIT_DATA_LENGTH 相同。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

ExitName 长度 (MQLONG)

此字段指定出口名称列表中由 *MsgExitPtr*，*SendExitPtr* 和 *ReceiveExitPtr* 字段寻址的每个名称的长度 (以字节计)。

此长度不一定与 MQ_EXIT_NAME_LENGTH 相同。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

HdrComp 列表 [2] (MQLONG)

此字段指定通道支持的头数据压缩技术的列表。

该列表包含以下一个或多个值:

MQCOMPRESS_NONE

不执行头数据压缩。

MQCOMPRESS_SYSTEM

执行头数据压缩。

数组中未使用的值设置为 MQCOMPRESS_NOT_AVAILABLE。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_8，那么该字段不存在。

HeartbeatInterval (MQLONG)

此字段指定脉动信号流之间的时间 (以秒计)。

此字段的解释取决于通道类型，如下所示:

- 对于通道类型 MQCHT_SENDER，MQCHT_SERVER，MQCHT_RECEIVER MQCHT_REQUESTER，MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR，此字段是在传输队列上没有消息时从发送 MCA 传递的脉动信号流之间的时间 (以秒为单位)。这使接收 MCA 有机会停顿通道。为了有用，*HeartbeatInterval* 必须小于 *DiscInterval*。
- 对于 MQCD "共享对话" 字段设置为零的 MQCHT_CLNTCONN 或 MQCHT_SVRCONN 通道类型，此字段是当 MCA 代表客户机应用程序使用 MQGMO_WAIT 选项发出 MQGET 调用时，从服务器 MCA 传递的脉动信号流之间的时间 (以秒为单位)。这允许服务器 MCA 处理在使用 MQGMO_WAIT 的 MQGET 期间客户机连接失败的情况。
- 对于 MQCHT_CLNTCONN 或 MQCHT_SVRCONN 的通道类型，将 "MQCD 共享对话" 字段设置为非零值，此字段是没有发送或接收数据流时脉动信号流之间的时间 (以秒为单位)。这允许高效地停顿通道。

该值在 0 到 999 999 范围内。使用的值是在发送端和接收端指定的值中较大的值，除非在任一端指定了值 0，在这种情况下不会发生脉动信号交换。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

KeepAlive 时间间隔 (MQLONG)

此字段指定传递到通信堆栈的值，以用于通道的保持活动计时。

该值适用于 TCP/IP 和 SPX 通信协议，但并非所有实现都支持此参数。

该值在范围 0 到 99 999 之间；单位为秒。值为零表示未启用通道保持活动，尽管如果启用了 TCP/IP 保持活动（而不是通道保持活动），那么仍可能发生保持活动。以下特殊值也有效：

MQKAI_AUTO

自动。

此值指示根据协商的脉动信号间隔计算保持活动时间间隔，如下所示：

- 如果协商的脉动信号间隔大于零，那么使用的保持活动时间间隔是脉动信号间隔加 60 秒。
- 如果协商的脉动信号间隔为零，那么使用的保持活动时间间隔为零。
- 在 z/OS 上，当在队列管理器对象上指定 TCPKEEP (YES) 时，将发生 TCP/IP 保持活动。
- 在其他环境中，当在分布式排队配置文件的 TCP 节中指定了 KEEPALIVE=YES 参数时，将发生 TCP/IP keepalive。

此字段仅与 *TransportType* 为 MQXPT_TCP 或 MQXPT_SPX 的通道相关。

这是出口的输出字段。如果 *Version* 小于 MQCD_VERSION_7，那么该字段不存在。

LocalAddress (MQCHAR48)

此字段指定为用于出站通信的通道定义的本地 TCP/IP 地址。

如果没有为出站通信定义特定地址，那么此字段为空白。该地址可以选择包含端口号或端口号范围。此地址的格式为：

```
[ip-addr][(low-port[,high-port])]
```

其中，方括号 ([]) 表示可选信息，*ip-addr* 以 IPv4 点分十进制，IPv6 十六进制或字母数字格式指定，*low-port* 和 *high-port* 是用括号括起来的端口号。所有属性都是可选的。

出站通信的特定 IP 地址，端口或端口范围在不同 TCP/IP 堆栈上重新启动通道的恢复场景中很有用。

LocalAddress 的格式类似于 *ConnectionName*，但不能与之混淆。*LocalAddress* 指定本地通信的特征，而 *ConnectionName* 指定如何访问远程队列管理器。

此字段仅与 *TransportType* 为 MQXPT_TCP 且 *ChannelType* 为 MQCHT_SENDER，MQCHT_SERVER，MQCHT_REQUESTER，MQCHT_CLNTCONN，MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

此字段的长度由 MQ_LOCAL_ADDRESS_LENGTH 提供。如果 *Version* 小于 MQCD_VERSION_7，那么此字段不存在。

LongMCAUserIdLength (MQLONG)

此字段指定 *LongMCAUserIdPtr* 指向的完整 MCA 用户标识的长度（以字节计）。

此字段与 *ChannelType* 为 MQCHT_CLNTCONN 的通道无关。

这是出口的输出/输入字段。如果 *Version* 小于 MQCD_VERSION_6，那么该字段不存在。

LongMCAUserIdPtr (MQPTR)

此字段指定长 MCA 用户标识的地址。

如果 *LongMCAUserIdLength* 大于零，那么此字段是完整 MCA 用户标识的地址。完整标识的长度由 *LongMCAUserIdLength* 提供。MCA 用户标识的前 12 个字节也包含在 *MCAUserIdentifier* 字段中。

请参阅 *MCAUserIdentifier* 字段的描述以获取 MCA 用户标识的详细信息。

此字段与 *ChannelType* 为 MQCHT_SDR，MQCHT_SVR，MQCHT_CLNTCONN 或 MQCHT_CLUSSDR 的通道无关。

这是出口的输出/输入字段。如果 *Version* 小于 MQCD_VERSION_6，那么该字段不存在。

LongRemoteUserId 长度 (MQLONG)

此字段指定 *LongRemoteUserIdPtr* 指向的完整远程用户标识的长度 (以字节计)。

此字段仅与 *ChannelType* 为 MQCHT_CLNTCONN 或 MQCHT_SVRCONN 的通道相关。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_6, 那么该字段不存在。

LongRemoteUserIdPtr (MQPTR)

此字段指定长远程用户标识的地址。

如果 *LongRemoteUserIdLength* 大于零, 那么此标志是完整远程用户标识的地址。完整标识的长度由 *LongRemoteUserIdLength* 提供。远程用户标识的前 12 个字节也包含在字段 *RemoteUserIdentifier* 中。

请参阅 *RemoteUserIdentifier* 字段的描述以获取远程用户标识的详细信息。

此字段仅与 *ChannelType* 为 MQCHT_CLNTCONN 或 MQCHT_SVRCONN 的通道相关。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_6, 那么该字段不存在。

LongRetry 计数 (MQLONG)

此字段指定在 *ShortRetryCount* 指定的计数耗尽后使用的计数。

它指定在将错误记录到操作员之前, 按 *LongRetryInterval* 指定的时间间隔进一步尝试连接到远程机器的最大次数。

此字段仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

LongRetry 时间间隔 (MQLONG)

此字段指定在重新尝试与远程机器的连接之前等待的最大秒数。

如果通道必须等待变为活动状态, 那么可以延长重试之间的时间间隔。

此字段仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

MaxInstances (MQLONG)

此字段指定可启动的单个服务器连接通道的最大并发实例数。

此字段仅在服务器连接通道上使用。

该字段可以具有 0-999 999 999 范围内的值。值为 0 表示阻止所有客户机访问。

此字段的缺省值为 999 999 999。

如果此字段的值小于当前正在运行的服务器连接通道的实例数, 那么正在运行的实例不受影响。但是, 只有在足够多的现有实例停止运行之后, 才能启动新实例, 以使当前正在运行的实例数小于该字段的值。

MaxInstancesPerClient (MQLONG)

此字段指定可从单个客户机启动的单个服务器连接通道的最大并发实例数。

在此上下文中, 源自同一远程网络地址的连接被视为来自同一客户机。

此字段仅在服务器连接通道上使用。

该字段可以具有 0-999 999 999 范围内的值。值为 0 表示阻止所有客户机访问。

此字段的缺省值为 999 999 999。

如果此字段的值减小到小于当前从个别客户机运行的服务器连接通道实例数, 那么正在运行的实例不受影响。但是, 任何这些客户机的新实例都无法启动, 直到有足够的现有实例停止运行, 使得当前正在运行的实例 (源自尝试启动新实例的客户机) 的数目小于该字段的值。

MaxMsg 长度 (MQLONG)

此字段指定可在通道上传输的最大消息长度。

该值和远程通道的值相比较，实际的最大值是两个值中较小的一个。

MCAName (MQCHAR20)

此字段是保留字段。

此字段的值为空白。

此字段的长度由 MQ_MCA_NAME_LENGTH 提供。

MCASecurityId (MQBYTE40)

此字段指定 MCA 的安全标识。

此字段与 *ChannelType* 为 MQCHT_CLNTCONN 的通道无关。

以下特殊值指示没有安全标识：

MQSID_NONE

未指定安全标识。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQSID_NONE_ARRAY；此常量具有与 MQSID_NONE 相同的值，但是字符数组而不是字符串。

这是出口的输出/输入字段。此字段的长度由 MQ_SECURITY_ID_LENGTH 指定。如果 *Version* 小于 MQCD_VERSION_6，那么此字段不存在。

MCAType (MQLONG)

此字段指定消息通道代理程序的类型。

此字段仅与 *ChannelType* 为 MQCHT_SENDER，MQCHT_SERVER，MQCHT_REQUESTER，MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

该值为下列其中之一：

MQMCAT_PROCESS

process.

消息通道代理程序作为一个单独的进程运行。

MQMCAT_THREAD

线程 (IBM i, UNIX 和 Windows)。

消息通道代理程序作为一个单独的线程运行。

当版本小于 MQCD_VERSION_2 时，此字段不存在。

MCAUserIdentifier (MQCHAR12)

此字段指定消息通道代理程序 (MCA) 的用户标识。

此字段使用 MCA 用户标识的前 12 个字节，并且可以由安全代理程序设置。

有两个字段包含 MCA 用户标识：

- *MCAUserIdentifier* 包含 MCA 用户标识的前 12 个字节，如果标识短于 12 个字节，那么将用空白填充。*MCAUserIdentifier* 可以为空。
- *LongMCAUserIdPtr* 指向完整的 MCA 用户标识，该标识的长度可能超过 12 个字节。其长度由 *LongMCAUserIdLength* 给出。完整标识不包含尾部空格，并且不是以 null 结束的。如果标识为空，那么 *LongMCAUserIdLength* 为零，并且未定义 *LongMCAUserIdPtr* 的值。

注：如果 *Version* 小于 MQCD_VERSION_6，那么 *LongMCAUserIdPtr* 不存在。

如果 MCA 用户标识为非空白，那么它指定消息通道代理程序用于授权访问 WebSphere MQ 资源的用户标识。对于通道类型 MQCHT_REQUESTER，MQCHT_RECEIVER 和 MQCHT_CLUSRCVR，如果 *PutAuthority* 为 MQPA_DEFAULT，那么这是用于对目标队列的放置操作进行授权检查的用户标识。

如果 MCA 用户标识为空，那么消息通道代理程序将使用其缺省用户标识。

MCA 用户标识可由安全出口设置，以指示消息通道代理程序必须使用的用户标识。出口可以更改 *MCAUserIdentifier* 或 *LongMCAUserIdPtr* 所指向的字符串。如果两者都已更改但彼此不同，那么 MCA 将使用 *LongMCAUserIdPtr* 而不是 *MCAUserIdentifier*。如果出口更改了由 *LongMCAUserIdPtr* 寻址的字符串的长度，那么必须相应地设置 *LongMCAUserIdLength*。如果出口增加了标识的长度，那么出口必须分配所需长度的存储器，将该存储器设置为所需标识，并将该存储器的地址放在 *LongMCAUserIdPtr* 中。当稍后使用 MQXR_TERM 原因调用该出口时，该出口负责释放该存储器。

对于 *ChannelType* 为 MQCHT_SVRCONN 的通道，如果通道定义中的 *MCAUserIdentifier* 为空，那么会将从客户机传输的任何用户标识复制到其中。此用户标识 (在服务器上的安全出口进行任何修改之后) 是假定客户机应用程序正在运行的用户标识。

MCA 用户标识与 *ChannelType* 为 MQCHT_SDR，MQCHT_SVR，MQCHT_CLNTCONN 和 MQCHT_CLUSSDR 的通道无关。

这是出口的输入/输出字段。此字段的长度由 MQ_USER_ID_LENGTH 给出。当 *Version* 小于 MQCD_VERSION_2 时，此字段不存在。

ModeName (MQCHAR8)

此字段指定 LU 6.2 方式名。

仅当传输协议 (*TransportType*) 为 MQXPT_LU62 并且 *ChannelType* 不是 MQCHT_SVRCONN 或 MQCHT_RECEIVER 时，此字段才相关。

此字段始终为空白。而是包含在通信端对象中。

此字段的长度由 MQ_MODE_NAME_LENGTH 指定。

MsgComp 列表 [16] (MQLONG)

此字段指定通道支持的消息数据压缩技术的列表。

该列表包含以下一个或多个值:

MQCOMPRESS_NONE

不执行消息数据压缩。

MQCOMPRESS_RLE

使用运行长度编码执行消息数据压缩。

MQCOMPRESS_ZLIBFAST

使用 zlib 压缩技术来执行消息数据压缩。推荐使用快速压缩时间。

MQCOMPRESS_ZLIBHIGH

使用 zlib 压缩技术来执行消息数据压缩。推荐使用高级压缩。

数组中未使用的值设置为 MQCOMPRESS_NOT_AVAILABLE。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_8，那么该字段不存在。

MsgExit (MQCHARn)

此字段指定通道消息出口名称。

如果此名称非空白，那么将在以下时间调用该出口:

- 在从传输队列 (发送方或服务器) 检索消息之后，或者在将消息放入目标队列 (接收方或请求者) 之前，立即执行此操作。

将为出口提供整个应用程序消息和传输队列头以进行修改。

- 在通道初始化和终止时。

此字段与 *ChannelType* 为 MQCHT_SVRCONN 或 MQCHT_CLNTCONN 的通道无关; 从不对此类通道调用消息出口。

有关此字段在各种环境中的内容的描述，请参阅第 922 页的『MQCD-通道定义』。

此字段的长度由 MQ_EXIT_NAME_LENGTH 指定。

注: 此常量的值特定于环境。

MsgExitPtr (MQPTR)

此字段指定第一个 *MsgExit* 字段的地址。

如果 *MsgExitsDefined* 大于零，那么此地址是链中每个通道消息出口的名称列表的地址。

每个名称都在长度为 *ExitNameLength* 的字段中，用空格填充到右边。有 *MsgExitsDefined* 个字段彼此相邻-每个出口一个字段。

将保留出口对这些名称所作的任何更改，尽管消息通道出口不执行显式操作-它不会更改调用的出口。

如果 *MsgExitsDefined* 为零，那么此字段为空指针。

在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

MsgExits 已定义 (MQLONG)

此字段指定在链中定义的通道消息出口数。

大于或等于零。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

MsgRetry 计数 (MQLONG)

此字段指定第一次尝试失败后 MCA 尝试放入消息的次数。

此字段指示当第一个 MQOPEN 或 MQPUT 失败且完成代码为 MQCC_FAILED 时，MCA 尝试打开或放入操作的次数。此属性的效果取决于 *MsgRetryExit* 是空白还是非空白：

- 如果 *MsgRetryExit* 为空，那么 *MsgRetryCount* 属性将控制 MCA 是否尝试重试。如果属性值为零，那么不会尝试任何重试。如果属性值大于零，那么将按 *MsgRetryInterval* 属性给定的时间间隔尝试重试。

仅针对以下原因码尝试重试：

- MQRC_PAGESET_FULL
- MQRC_PUT_禁止
- MQRC_Q_FULL

对于其他原因码，MCA 将立即执行其正常故障处理，而不重试失败消息。

- 如果 *MsgRetryExit* 为非空白，那么 *MsgRetryCount* 属性不会影响 MCA；相反，它是消息重试出口，用于确定尝试重试的次数以及时间间隔；即使 *MsgRetryCount* 属性为零，也会调用该出口。

MsgRetryCount 属性可用于 MQCD 结构中的出口，但不需要使用该属性的出口-重试将无限期继续，直到该出口在 MQCXP 的 *ExitResponse* 字段中返回 MQXCC_SUPPRESS_FUNCTION 为止。

此字段仅与 *ChannelType* 为 MQCHT_REQUESTER，MQCHT_RECEIVER 或 MQCHT_CLUSRCVR 的通道相关。

当 *Version* 小于 MQCD_VERSION_3 时，此字段不存在。

MsgRetry 退出 (MQCHARn)

此字段指定通道消息重试出口名称。

消息重试出口是 MCA 从 MQOPEN 或 MQPUT 调用接收到完成代码 MQCC_FAILED 时调用的出口。出口的目的地是指定 MCA 在重试 MQOPEN 或 MQPUT 操作之前等待的时间间隔。或者，可以将出口设置为不重试该操作。

将对完成代码为 MQCC_FAILED 的所有原因码调用该出口-该出口的设置确定希望 MCA 重试的原因码，尝试次数以及时间间隔。

当不再尝试该操作时，MCA 将执行其正常故障处理；此处理包括生成异常报告消息（如果由发送方指定），并将原始消息放入死信队列或废弃消息（根据发送方是否指定了 MQRO_DEAD_LETTER_Q 或 MQRO_DISCARD_MSG）。涉及死信队列的故障（例如，死信队列已满）不会导致调用消息重试出口。

如果出口名称为非空白，那么将在以下时间调用该出口：

- 在执行等待之前立即尝试再次传递消息

- 在通道初始化和终止时

有关此字段在各种环境中的内容的描述, 请参阅 [第 922 页的『MQCD-通道定义』](#)。

此字段仅与 *ChannelType* 为 MQCMT_REQUESTER, MQCMT_RECEIVER 或 MQCMT_CLUSRCVR 的通道相关。

此字段的长度由 MQ_EXIT_NAME_LENGTH 指定。

注: 此常量的值特定于环境。

当 *Version* 小于 MQCD_VERSION_3 时, 此字段不存在。

MsgRetry 时间间隔 (MQLONG)

此字段指定重试打开或放入操作的最小时间间隔 (以毫秒为单位)。

此属性的效果取决于 *MsgRetryExit* 是空白还是非空白:

- 如果 *MsgRetryExit* 为空, 那么 *MsgRetryInterval* 属性指定当第一个 MQOPEN 或 MQPUT 失败且完成代码为 MQCC_FAILED 时, MCA 在重试消息之前等待的最短时间段。值为零表示将在上次尝试后尽快执行重试。仅当 *MsgRetryCount* 大于零时, 才会执行重试。

如果消息重试出口在 MQCXP 的 *MsgRetryInterval* 字段中返回无效值, 那么此属性也将用作等待时间。

- 如果 *MsgRetryExit* 不为空, 那么 *MsgRetryInterval* 属性不会影响 MCA; 相反, 它是确定 MCA 等待时间的消息重试出口。 *MsgRetryInterval* 属性可用于 MQCD 结构中的出口, 但不需要该出口来实现该出口。

该值在范围 0 到 999 999 999 之间。

此字段仅与 *ChannelType* 为 MQCMT_REQUESTER, MQCMT_RECEIVER 或 MQCMT_CLUSRCVR 的通道相关。

当 *Version* 小于 MQCD_VERSION_3 时, 此字段不存在。

如果 *Version* 小于 MQCD_VERSION_4, 那么此结构中的以下字段不存在。

MsgRetryUserData (MQCHAR32)

此字段指定通道消息重试出口用户数据。

此数据将传递到 *ChannelExitParms* 参数的 *ExitData* 字段中的通道消息重试出口 (请参阅 MQ_CHANNEL_EXIT)。

此字段最初包含在通道定义中设置的数据。但是, 在此 MCA 实例的生存期内, 任何类型的出口对此字段内容所作的任何更改都将由 MCA 保留, 并使此 MCA 实例的后续出口调用 (无论类型如何) 可视。此类更改不会影响其他 MCA 实例使用的通道定义。可以使用任何字符 (包括二进制数据)。

此字段仅与 *ChannelType* 为 MQCMT_REQUESTER, MQCMT_RECEIVER 或 MQCMT_CLUSRCVR 的通道相关。

此字段的长度由 MQ_EXIT_DATA_LENGTH 指定。当 *Version* 小于 MQCD_VERSION_3 时, 此字段不存在。

此字段在 WebSphere MQ for IBM i 中不相关。

MsgUser 数据 (MQCHAR32)

此字段指定通道消息出口用户数据。

此数据将传递到 *ChannelExitParms* 参数的 *ExitData* 字段中的通道消息出口 (请参阅 MQ_CHANNEL_EXIT)。

此字段最初包含在通道定义中设置的数据。但是, 在此 MCA 实例的生存期内, 任何类型的出口对此字段内容所作的任何更改都将由 MCA 保留, 并使此 MCA 实例的后续出口调用 (无论类型如何) 可视。此类更改不会影响其他 MCA 实例使用的通道定义。可以使用任何字符 (包括二进制数据)。

此字段的长度由 MQ_EXIT_DATA_LENGTH 指定。

此字段在 WebSphere MQ for IBM i 中不相关。

MsgUserDataPtr (MQPTR)

此字段指定第一个 *MsgUserData* 字段的地址。

如果 *MsgExitsDefined* 大于零, 那么此地址是链中每个通道消息出口的用户数据项列表的地址。

每个用户数据项都在长度为 *ExitDataLength* 的字段中, 用空白填充到右侧。有 *MsgExitsDefined* 个字段彼此相邻-每个出口一个字段。如果定义的用户数据项数量小于出口名称数量, 那么未定义的用户数据项将设置为空白。相反, 如果定义的用户数据项数量大于出口名称数量, 那么将忽略多余的用户数据项, 并且不会向出口显示这些数据项。

将保留出口对这些值所作的任何更改。这允许一个出口将信息传递到另一个出口。不会对任何更改执行验证, 例如, 如果需要, 可以将二进制数据写入这些字段。

如果 *MsgExitsDefined* 为零, 那么此字段为空指针。

在编程语言不支持指针数据类型的平台上, 此字段声明为适当长度的字节字符串。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4, 那么该字段不存在。

NetworkPriority (MQLONG)

此字段指定通道的网络连接的优先级。

当特定目标的多条路径可用时, 将选择具有最高优先级的路径。该值在 0 到 9 范围内; 0 是最低优先级。

此字段仅与 *ChannelType* 为 MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_5, 那么此字段不存在。

如果 *Version* 小于 MQCD_VERSION_6, 那么此结构中的以下字段不存在。

NonPersistentMsgSpeed (MQLONG)

此字段指定非持久消息通过通道的速度。

此字段仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

该值为下列其中之一:

MQNPMS_NORMAL

正常速度。

如果通道定义为 MQNPMS_NORMAL, 那么非持久消息将以正常速度通过通道。这具有以下优点: 如果发生通道故障, 那么不会丢失这些消息。此外, 同一传输队列上的持久消息和非持久消息保持它们之间的相对顺序。

MQNPMS_FAST

速度很快。

如果将通道定义为 MQNPMS_FAST, 那么非持久消息将以快速速度通过该通道。这将提高通道的吞吐量, 但意味着如果发生通道故障, 将丢失非持久消息。此外, 非持久消息也可能在等待同一传输队列的持久消息之前跳转, 即, 相对于持久消息, 非持久消息的顺序不会保持。但是, 将保持非持久消息相对于彼此的顺序。同样, 将保持持久消息相对于彼此的顺序。

密码 (MQCHAR12)

此字段指定尝试启动与远程消息通道代理程序的安全 SNA 会话时消息通道代理程序使用的密码。

此字段只能在 UNIX 系统和 Windows 上为非空白, 并且仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER 或 MQCHT_CLNTCONN 的通道相关。在 z/OS 上, 此字段不相关。

此字段的长度由 MQ_PASSWORD_LENGTH 给出。但是, 仅使用前 10 个字符。

如果 *Version* 小于 MQCD_VERSION_2, 那么此字段不存在。

PropertyControl (MQLONG)

此字段指定将消息发送到 V6 或先前队列管理器 (不了解属性描述符概念的队列管理器) 时, 消息的属性将发生什么情况。

值可以是：

MQPROP_COMPATIBILITY

如果消息包含前缀为 **mcd.**、**jms.**、**usr.** 或 **mqext.** 的属性，那么所有消息属性都将通过 MQRFH2 头传递到应用程序。否则，将废弃消息的所有属性（消息描述符（或扩展）中包含的属性除外），并且应用程序无法再访问这些属性。

此值是缺省值；它允许期望 JMS 相关属性位于消息数据中的 MQRFH2 头中的应用程序继续未修改地工作。

MQPROP_NONE

在将消息发送到远程队列管理器之前，将从消息中除去消息的所有属性（消息描述符（或扩展）中的属性除外）。

MQPROP_ALL

将消息发送到远程队列管理器时，该消息的所有属性都包含在该消息中。除消息描述符（或扩展）中的属性外，其他属性将放置在消息数据的一个或多个 MQRFH2 头中。

此属性适用于发送方，服务器，集群发送方和集群接收方通道。

[第 112 页的『MQIA_* \(整数属性选择器\)』](#)

[第 149 页的『MQPROP_* \(队列和通道属性控制值以及最大属性长度\)』](#)

PutAuthority (MQLONG)

此字段指定是否使用与消息关联的上下文信息中的用户标识来建立将消息放入目标队列的权限。

此字段仅与 *ChannelType* 为 MQCHT_REQUESTER、MQCHT_RECEIVER 或 MQCHT_CLUSRCVR 的通道相关。它是下列项之一：

MQPA_DEFAULT

使用缺省用户标识。

MQPA_CONTEXT

使用上下文用户标识。

MQPA_ALTERNATE_OR_MCA

将使用消息描述符的 UserIdentifier 字段中的用户标识。不使用从网络接收的任何用户标识。此值仅在 z/OS 上受支持。

MQPA_ONLY_MCA

将使用缺省用户标识。不使用从网络接收的任何用户标识。此值仅在 z/OS 上受支持。

QMgrName (MQCHAR48)

此字段指定出口可以连接到的队列管理器的名称。

对于具有 *ChannelType* 而不是 MQCHT_CLNTCONN 的通道，此字段是出口可连接到的队列管理器的名称，在 UNIX、Linux 和 Windows 系统上，该队列管理器始终为非空白。

此字段的长度由 MQ_Q_MGR_NAME_LENGTH 提供。

ReceiveExit (MQCHARn)

此字段指定通道接收出口名称。

如果此名称非空白，那么将在以下时间调用该出口：

- 在处理接收到的网络数据之前。
为出口提供了接收到的完整传输缓冲区。可以根据需要修改缓冲区的内容。
- 在通道初始化和终止时。

有关此字段在各种环境中的内容的描述，请参阅 [第 922 页的『MQCD-通道定义』](#)。

此字段的长度由 MQ_EXIT_NAME_LENGTH 指定。

注：此常量的值特定于环境。

ReceiveExitPtr (MQPTR)

此字段指定第一个 *ReceiveExit* 字段的地址。

如果 *ReceiveExitsDefined* 大于零，那么此地址是链中每个通道接收出口的名称列表的地址。

每个名称都在长度为 *ExitNameLength* 的字段中，用空格填充到右边。有 *ReceiveExitsDefined* 个字段彼此相邻-每个出口一个字段。

将保留出口对这些名称所作的任何更改，尽管消息通道出口不执行显式操作-它不会更改调用的出口。

如果 *ReceiveExitsDefined* 为零，那么此字段为空指针。

在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

ReceiveExits 已定义 (MQLONG)

此字段指定在链中定义的通道接收出口数。

大于或等于零。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

ReceiveUser 数据 (MQCHAR32)

此通道指定通道接收出口用户数据。

此数据将传递到 *ChannelExitParms* 参数的 *ExitData* 字段中的通道接收出口 (请参阅 MQ_CHANNEL_EXIT)。

此字段最初包含在通道定义中设置的数据。但是，在此 MCA 实例的生存期内，任何类型的出口对此字段内容所作的任何更改都将由 MCA 保留，并使此 MCA 实例的后续出口调用 (无论类型如何) 可视。这适用于不同对话上的出口。此类更改不会影响其他 MCA 实例使用的通道定义。可以使用任何字符 (包括二进制数据)。

此字段的长度由 MQ_EXIT_DATA_LENGTH 指定。

此字段在 WebSphere MQ for IBM i 中不相关。

如果 *Version* 小于 MQCD_VERSION_2，那么此结构中的以下字段不存在。

ReceiveUserDataPtr (MQPTR)

此字段指定第一个 *ReceiveUserData* 字段的地址。

如果 *ReceiveExitsDefined* 大于零，那么此地址是链中每个通道接收出口的用户数据项列表的地址。

每个用户数据项都在长度为 *ExitDataLength* 的字段中，用空白填充到右侧。有 *ReceiveExitsDefined* 个字段彼此相邻-每个出口一个字段。如果定义的用户数据项数量小于出口名称数量，那么未定义的用户数据项将设置为空白。相反，如果定义的用户数据项数量大于出口名称数量，那么将忽略多余的用户数据项，并且不会向出口显示这些数据项。

将保留出口对这些值所作的任何更改。这允许一个出口将信息传递到另一个出口。不会对任何更改执行验证，例如，如果需要，可以将二进制数据写入这些字段。

如果 *ReceiveExitsDefined* 为零，那么此字段为空指针。

在编程语言不支持指针数据类型的平台上，此字段声明为适当长度的字节字符串。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4，那么该字段不存在。

如果 *Version* 小于 MQCD_VERSION_5，那么此结构中的以下字段不存在。

RemotePassword (MQCHAR12)

此字段指定来自合作伙伴的密码。

仅当 *ChannelType* 为 MQCHT_CLNTCONN 或 MQCHT_SVRCONN 时，此字段才包含有效信息。

- 对于 MQCHT_CLNTCONN 通道的安全出口，此密码是从环境中获取的密码。该出口可以选择将其发送到服务器上的安全出口。

- 对于 MQCMT_SVRCONN 通道的安全出口，如果没有客户机安全出口，那么此字段可能包含从客户机环境获取的密码。出口可以使用此密码来验证 *RemoteUserIdentifier* 中的用户标识。

如果客户机上存在安全出口，那么可以在客户机的安全流中获取此信息。

此字段的长度由 MQ_PASSWORD_LENGTH 给出。如果 *Version* 小于 MQCD_VERSION_2，那么此字段不存在。

RemoteSecurity 标识 (MQBYTE40)

此字段指定远程用户的安全标识。

此字段仅与 *ChannelType* 为 MQCMT_CLNTCONN 或 MQCMT_SVRCONN 的通道相关。

以下特殊值指示没有安全标识：

MQSID_NONE

未指定安全标识。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQSID_NONE_ARRAY；此常量具有与 MQSID_NONE 相同的值，但是字符数组而不是字符串。

这是出口的输入字段。此字段的长度由 MQ_SECURITY_ID_LENGTH 指定。如果 *Version* 小于 MQCD_VERSION_6，那么此字段不存在。

如果 *Version* 小于 MQCD_VERSION_7，那么此结构中的以下字段不存在。

RemoteUser 标识 (MQCHAR12)

此字段指定来自合作伙伴的用户标识的前 12 个字节。

有两个字段包含远程用户标识：

- *RemoteUserIdentifier* 包含远程用户标识的前 12 个字节，如果该标识短于 12 个字节，那么将使用空白进行填充。*RemoteUserIdentifier* 可以为空。
- *LongRemoteUserIdPtr* 指向完整的远程用户标识，该标识的长度可能超过 12 个字节。其长度由 *LongRemoteUserIdLength* 给出。完整标识不包含尾部空格，并且不是以 null 结束的。如果标识为空，那么 *LongRemoteUserIdLength* 为零，并且未定义 *LongRemoteUserIdPtr* 的值。

如果 *Version* 小于 MQCD_VERSION_6，那么 *LongRemoteUserIdPtr* 不存在。

远程用户标识仅与 *ChannelType* 为 MQCMT_CLNTCONN 或 MQCMT_SVRCONN 的通道相关。

- 对于 MQCMT_CLNTCONN 通道上的安全出口，此值是已从环境中获取的用户标识。该出口可以选择将其发送到服务器上的安全出口。
- 对于 MQCMT_SVRCONN 通道上的安全出口，如果没有客户机安全出口，那么此字段可能包含从客户机环境获取的用户标识。该出口可能会验证此用户标识（可能使用 *RemotePassword* 中的密码）并更新 *MCAUserIdentifier* 中的值。

如果客户机上存在安全出口，那么可以在客户机的安全流中获取此信息。

此字段的长度由 MQ_USER_ID_LENGTH 给出。如果 *Version* 小于 MQCD_VERSION_2，那么此字段不存在。

SecurityExit (MQCHARn)

此字段指定通道安全出口名称。

如果此名称非空白，那么将在以下时间调用该出口：

- 建立通道后立即调用。
在传输任何消息之前，出口将有机会促使安全性流来验证连接权限。
- 收到对安全消息流的响应时。
将从远程机器上的远程处理器接收的任何安全消息流提供给出口。
- 在通道初始化和终止时。

有关此字段在各种环境中的内容的描述, 请参阅第 922 页的『MQCD-通道定义』。

此字段的长度由 MQ_EXIT_NAME_LENGTH 指定。

注: 此常量的值特定于环境。

SecurityUser 数据 (MQCHAR32)

此通道指定通道安全性出口用户数据。

此数据将传递到 *ChannelExitParms* 参数的 *ExitData* 字段中的通道安全性出口 (请参阅 MQ_CHANNEL_EXIT)。

此字段最初包含在通道定义中设置的数据。但是, 在此 MCA 实例的生存期内, 任何类型的出口对此字段内容所作的任何更改都将由 MCA 保留, 并使此 MCA 实例的后续出口调用 (无论类型如何) 可视。这适用于不同对话上的出口。此类更改不会影响其他 MCA 实例所使用的通道定义。可以使用任何字符 (包括二进制数据)。

此字段的长度由 MQ_EXIT_DATA_LENGTH 指定。

此字段在 WebSphere MQ for IBM i 中不相关。

SendExit (MQCHARn)

此字段指定通道发送出口名称。

如果此名称非空白, 那么将在以下时间调用该出口:

- 在网络上发送数据之前, 请立即执行此操作。
在传输之前, 将为出口提供完整的传输缓冲区。可以根据需要修改缓冲区的内容。
- 在通道初始化和终止时。

有关此字段在各种环境中的内容的描述, 请参阅第 922 页的『MQCD-通道定义』。

此字段的长度由 MQ_EXIT_NAME_LENGTH 指定。

注: 此常量的值特定于环境。

SendExitPtr (MQPTR)

此字段指定第一个 *SendExit* 字段的地址。

如果 *SendExitsDefined* 大于零, 那么此地址是链中每个通道发送出口的名称列表的地址。

每个名称都在长度为 *ExitNameLength* 的字段中, 用空格填充到右边。有 *SendExitsDefined* 个字段彼此相邻-每个出口一个字段。

将保留出口对这些名称所作的任何更改, 尽管消息发送出口不执行显式操作-它不会更改调用的出口。

如果 *SendExitsDefined* 为零, 那么此字段为空指针。

在编程语言不支持指针数据类型的平台上, 此字段声明为适当长度的字节字符串。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4, 那么该字段不存在。

SendExits 已定义 (MQLONG)

此字段指定在链中定义的通道发送出口数。

大于或等于零。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4, 那么该字段不存在。

SendUser 数据 (MQCHAR32)

此字段指定通道发送出口用户数据。

此数据将传递到 *ChannelExitParms* 参数的 *ExitData* 字段中的通道发送出口 (请参阅 MQ_CHANNEL_EXIT)。

此字段最初包含在通道定义中设置的数据。但是, 在此 MCA 实例的生存期内, 任何类型的出口对此字段内容所作的任何更改都将由 MCA 保留, 并使此 MCA 实例的后续出口调用 (无论类型如何) 可视。这适用于不

同对话上的出口。此类更改不会影响其他 MCA 实例使用的通道定义。可以使用任何字符 (包括二进制数据)。

此字段的长度由 MQ_EXIT_DATA_LENGTH 指定。

此字段在 WebSphere MQ for IBM i 中不相关。

SendUserDataPtr (MQPTR)

此字段指定 *SendUserData* 字段的地址。

如果 *SendExitsDefined* 大于零, 那么此地址是链中每个通道消息出口的用户数据项列表的地址。

每个用户数据项都在长度为 *ExitDataLength* 的字段中, 用空白填充到右侧。有 *MsgExitsDefined* 个字段彼此相邻-每个出口一个字段。如果定义的用户数据项数量小于出口名称数量, 那么未定义的用户数据项将设置为空白。相反, 如果定义的用户数据项数量大于出口名称数量, 那么将忽略多余的用户数据项, 并且不会向出口显示这些数据项。

将保留出口对这些值所作的任何更改。这允许一个出口将信息传递到另一个出口。不会对任何更改执行验证, 例如, 如果需要, 可以将二进制数据写入这些字段。

如果 *SendExitsDefined* 为零, 那么此字段为空指针。

在编程语言不支持指针数据类型的平台上, 此字段声明为适当长度的字节字符串。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_4, 那么该字段不存在。

SeqNumber 回绕 (MQLONG)

此字段指定允许的最大消息序号。

当达到此值时, 序号将换行以在 1 处再次开始。

此值不可协商, 并且必须在本地和远程通道定义中都匹配。

此字段与 *ChannelType* 为 MQCHT_SVRCONN 或 MQCHT_CLNTCONN 的通道无关。

SharingConversations (MQLONG)

此字段指定可以共享与此通道关联的通道实例的最大对话数。

此字段用于客户机连接和服务器连接通道。

值 0 表示通道在 WebSphere MQ V 7.0 之前的版本中按以下属性操作:

- 对话共享
- 提前读取
- STOP CHANNEL(<channelname>) MODE(QUIESCE)
- 正在发出脉动信号
- 客户机异步耗用

值 1 是 WebSphere MQ V7.0 行为的最小值。虽然通道实例上仅允许一个对话, 但仍提供了预读, 异步使用以及 CLNTCONN-SVRCONN 脉动信号传递和静默通道停止的 V 7 行为。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_9, 那么它不存在。

此字段的缺省值为 10。

注: *MaxInstances* 和 *MaxInstancesPerClient* 应用于通道的限制会限制通道实例数, 而不是限制可能共享这些实例的对话数。

ShortConnection 名称 (MQCHAR20)

此字段指定连接名称的前 20 个字节。

如果 *Version* 字段为 MQCD_VERSION_1, 那么 *ShortConnectionName* 包含完整连接名称。

如果 *Version* 字段为 MQCD_VERSION_2 或更高版本, 那么 *ShortConnectionName* 包含连接名称的前 20 个字符。完整连接名称由 *ConnectionName* 字段提供; *ShortConnectionName* 和 *ConnectionName* 的前 20 个字符完全相同。

请参阅 *ConnectionName* 以获取此字段内容的详细信息。

注: 对于 MQCD_VERSION_2 和后续版本的 MQCD, 此字段的名称已更改; 此字段先前称为 *ConnectionName*。

此字段的长度由 MQ_SHORT_CONN_NAME_LENGTH 指定。

ShortRetry 计数 (MQLONG)

此字段指定尝试连接到远程机器的最大次数。

此字段是在使用 (通常较长) *LongRetryCount* 和 *LongRetryInterval* 之前, 按 *ShortRetryInterval* 指定的时间间隔尝试连接到远程机器的最大次数。

此字段仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

ShortRetry 时间间隔 (MQLONG)

此字段指定在重新尝试与远程机器的连接之前等待的最大秒数。

如果通道必须等待变为活动状态, 那么重试之间的时间间隔可能会延长。

此字段仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR 或 MQCHT_CLUSRCVR 的通道相关。

SSLCipherSpec (MQCHAR32)

此字段指定使用 SSL 时正在使用的密码规范。

如果 *SSLCipherSpec* 为空, 那么表示通道未使用 SSL。如果不为空, 那么此字段包含一个字符串, 用于指定正在使用的 *CipherSpec*。

此参数对所有通道类型都有效。它在 AIX, HP-UX, Linux, IBM i, Solaris, Windows 和 z/OS 上受支持。它仅对传输类型 (TRPTYPE) 为 TCP 的通道类型有效。

这是出口的输入字段。此字段的长度由 MQ_SSL_CIPHER_SPEC_LENGTH 提供。如果 *Version* 小于 MQCD_VERSION_7, 那么该字段不存在。

SSLClientAuth (MQLONG)

此字段指定是否需要 SSL 客户机认证。

此字段仅与 SVRCONN 通道定义相关。

它是下列其中一个值:

MQSCA_REQUIRED

需要客户机认证。

MQSCA_OPTIONAL

客户机认证可选。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_7, 那么该字段不存在。

SSLPeerName 长度 (MQLONG)

此字段指定 *SSLPeerNamePtr* 所指向的 SSL 对等名称的长度 (以字节计)。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_7, 那么该字段不存在。

SSLPeerNamePtr (MQPTR)

此字段指定 SSL 对等名称的地址。

在成功进行 SSL 握手期间接收到证书时, 会将证书主体集的专有名称复制到接收证书的通道末尾的 *SSLPeerNamePtr* 所访问的 MQCD 字段中。如果此值存在于本地用户的通道定义中, 那么它将覆盖通道的 *SSLPeerName* 值。如果在通道的此端指定了安全出口, 那么它将从 MQCD 中的对等证书接收专有名称。

这是出口的输入字段。如果 *Version* 小于 MQCD_VERSION_7, 那么该字段不存在。

注: 在 WebSphere MQ v7.1 发行版之前构造的安全出口应用程序可能需要更新。有关更多信息, 请参阅 [通道安全出口程序](#)。

StrucLength (MQLONG)

此字段指定 MQCD 结构的长度 (以字节计)。

长度不包括结构中包含的指针字段所寻址的任何字符串。该值为下列其中之一:

MQCD_LENGTH_4

version-4 通道定义结构的长度。

MQCD_LENGTH_5

version-5 通道定义结构的长度。

MQCD_LENGTH_6

version-6 通道定义结构的长度。

MQCD_LENGTH_7

version-7 通道定义结构的长度。

MQCD_LENGTH_8

version-8 通道定义结构的长度。

MQCD_LENGTH_9

version-9 通道定义结构的长度。

以下常量指定当前版本的长度:

MQCD_CURRENT_LENGTH

通道定义结构的当前版本的长度。

注: 这些常量具有特定于环境的值。

如果 *Version* 小于 MQCD_VERSION_4, 那么该字段不存在。

TpName (MQCHAR64)

此字段指定 LU 6.2 事务程序名。

仅当传输协议 (*TransportType*) 为 MQXPT_LU62 并且 *ChannelType* 不是 MQCHT_SVRCONN 或 MQCHT_RECEIVER 时, 此字段才相关。

在通信端对象中包含信息的平台上, 此字段始终为空白。

此字段的长度由 MQ_TP_NAME_LENGTH 指定。

TransportType (MQLONG)

此字段指定要使用的传输协议。

如果从另一端启动通道, 那么不会检查该值。

它是下列其中一个值:

MQXPT_LU62

LU 6.2 传输协议。

MQXPT_TCP

TCP/IP 传输协议。

MQXPT_NETBIOS

NetBIOS 传输协议。

此值在以下环境中受支持: Windows。

MQXPT_SPX

SPX 传输协议。

此值在以下环境中受支持: Windows, 以及连接到这些系统的 WebSphere MQ 客户机。

UseDLQ (MQLONG)

此字段指定当通道无法传递消息时是否使用死信队列 (或未传递的消息队列)。

它可以包含下列其中一个值:

MQUSEDLQ_NO

通道无法传递的消息将被视为失败。根据 NPMSPEED 设置, 通道要么废弃消息, 要么通道结束。

MQUSEDLQ_YES

当 DEADQ 队列管理器属性提供死信队列的名称时, 将使用该队列, 否则行为与 NO 相同。缺省值是 YES。

UserIdentifier (MQCHAR12)

此字段指定尝试与远程消息通道代理程序启动安全 SNA 会话时消息通道代理程序使用的用户标识。

此字段只能在 UNIX 系统和 Windows 上为非空白, 并且仅与 *ChannelType* 为 MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER 或 MQCHT_CLNTCONN 的通道相关。在 z/OS 上, 此字段不相关。

此字段的长度由 MQ_USER_ID_LENGTH 给出。但是, 仅使用前 10 个字符。

当 *Version* 小于 MQCD_VERSION_2 时, 此字段不存在。

版本 (MQLONG)

Version 字段指定可以为结构设置的最高版本号。

该值取决于环境:

MQCD_VERSION_1

版本 1 通道定义结构。

MQCD_VERSION_2

版本 2 通道定义结构。

当前 IBM WebSphere MQ 产品未使用 V 2。

MQCD_VERSION_3

版本 3 通道定义结构。

在以下环境中, 可以将此字段设置为 MQSeries V 2 上的最高版本 3: HP Integrity NonStop Server, 以及未在其他位置列出的 UNIX and Linux 系统。

MQCD_VERSION_4

版本 4 通道定义结构。

当前 IBM WebSphere MQ 产品未使用 V 4。

MQCD_VERSION_5

版本 5 通道定义结构。

对于 MQSeries for OS/390 V 5 Release 2, 可以将此字段设置为 V 5 的最高版本。

MQCD_VERSION_6

版本 6 通道定义结构。

版本 6 不是任何现有 IBM WebSphere MQ 产品的当前 MQCD 结构版本。但是, 可以使用 MQCNO 结构的 ClientConnOffset 或 ClientConnPtr 字段将 V 6 MQCD 结构传递到 MQCONN。X。

在分布式平台上, V 6 是 MQCD_DEFAULT 和 MQCD_CLIENT_CONN_DEFAULT 初始化方法中的缺省版本。如果要引用 MQCD 的 MQCD_VERSION_7, MQCD_VERSION_8 或 MQCD_VERSION_9 字段, 请根据需要 MQCD **Version** 字段显式初始化为 MQCD_VERSION_7, MQCD_VERSION_8 或 MQCD_VERSION_9。

在 z/OS 上, MQCD_VERSION_7 是缺省值。

MQCD_VERSION_7

版本 7 通道定义结构。

在以下环境中，可以将此字段设置为 IBM WebSphere MQ Version 5.3 的最高版本 7: AIX, HP-UX, Solaris, Windows 以及 IBM WebSphere MQ for z/OS Version 5.3 和 Version 5.3.1。MQCD_VERSION_7 是 IBM WebSphere MQ for z/OS 版本的缺省值。

MQCD_VERSION_8

版本 8 通道定义结构。

在所有平台上的 IBM WebSphere MQ Version 6.0 上，可以将此字段设置为 V 8 的最高版本。

MQCD_VERSION_9

版本 9 通道定义结构。

在所有平台上的 IBM WebSphere MQ Version 7.0 和 IBM WebSphere MQ Version 7.0.1 上，可以将此字段设置为 V 9 的最高版本。

MQCD_VERSION_10

V10 通道定义结构。

在所有平台上的 IBM WebSphere MQ Version 7.1 和 IBM WebSphere MQ Version 7.5 上，可以将此字段设置为最高版本 10。

仅在结构的最新版本中存在的字段在字段的描述中标识为此类字段。以下常量指定当前版本的版本号：

MQCD_CURRENT_VERSION

MQCD_CURRENT_VERSION 中设置的值是正在使用的通道定义结构的当前版本。

MQCD_CURRENT_VERSION 的值取决于环境。它包含平台支持的最高值。

MQCD_CURRENT_VERSION 不用于初始化为不同编程语言提供的头，副本和包含文件中提供的缺省结构。Version 的缺省初始化取决于平台和发行版。

对于 IBM WebSphere MQ Version 7.0 和更高版本，头，副本和包含文件中的 MQCD 声明将初始化为 MQCD_VERSION_6。要使用其他 MQCD 字段，应用程序必须将版本号设置为 MQCD_CURRENT_VERSION。如果您正在编写可在多个环境之间移植的应用程序，那么必须选择在所有环境中支持的版本。

提示: 引入新版本的 MQCD 结构时，不会更改现有部件的布局。出口必须检查版本号。它必须等于或大于包含出口需要使用的字段的最低版本。

XmitQName (MQCHAR48)

此字段指定从中检索消息的传输队列的名称。

此字段仅与 *ChannelType* 为 MQCHT_SENDER 或 MQCHT_SERVER 的通道相关。

此字段的长度由 MQ_Q_NAME_LENGTH 提供。

C 声明

此声明是 MQCD 结构的 C 声明。

```
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 */

```

```

MQPTR      LongRemoteUserIdPtr;          /* identifier */
MQBYTE40   MCASecurityId;                /* Address of long remote */
MQBYTE40   RemoteSecurityId;            /* user 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 声明

此声明是 MQCD 结构的 COBOL 声明。

```

** 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-MAXMSGLLENGTH 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-NONPERSISTENTMSGSPPEED PIC S9(9) BINARY.
** Length of MQCD structure
15 MQCD-STRUCLLENGTH 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-MCASESECURITYID PIC X(40).
** Remote security identifier
 15 MQCD-REMOTESESECURITYID 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 **

```

RPG 声明 (ILE)

此声明是 MQCD 结构的 RPG 声明。

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 CDDDES 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 CDRCU 521 552
D* Ver:1 **
D* User identifier
D CDUID 553 564
D* Password
D CDPW 565 576
D* First 12 bytes of MCA user identifier
D CDAUI 577 588
D* Message channel agent type
D CDCAT 589 592I 0
D* Connection name
D CDCON 593 848
D CDCN2 849 856
D* First 12 bytes of user identifier from partner
D CDRUI 857 868
D* Password from partner
D CDRPW 869 880
D* Ver:2 **
D* Channel message retry exit name
D CDMRX 881 900
D* Channel message retry exit user data
D CDMRD 901 932
D* Number of times MCA will try to put the message, after first
D* attempt has failed
D CDMRC 933 936I 0

```

```

D* Minimum interval in milliseconds after which the open or put
D* operation will be retried
D CDMRI 937 940I 0
D* Ver:3 **
D* Time in seconds between heartbeat flows
D CDHBI 941 944I 0
D* Batch duration
D CDBI 945 948I 0
D* Speed at which nonpersistent messages are sent
D CDNPM 949 952I 0
D* Length of MQCD structure
D CDLEN 953 956I 0
D* Length of exit name
D CDXNL 957 960I 0
D* Length of exit user data
D CDXDL 961 964I 0
D* Number of message exits defined
D CDMXD 965 968I 0
D* Number of send exits defined
D CDSXD 969 972I 0
D* Number of receive exits defined
D CDRXD 973 976I 0
D* Address of first MsgExit field
D CDMXP 977 992*
D* Address of first MsgUserData field
D CDMUP 993 1008*
D* Address of first SendExit field
D CDSXP 1009 1024*
D* Address of first SendUserData field
D CDSUP 1025 1040*
D* Address of first ReceiveExit field
D CDRXP 1041 1056*
D* Address of first ReceiveUserData field
D CDRUP 1057 1072*
D* Ver:4 **
D* Address of a list of cluster names
D CDCLP 1073 1088*
D* Number of clusters to which the channel belongs
D CDCLD 1089 1092I 0
D* Network priority
D CDNP 1093 1096I 0
D* Ver:5 **
D* Length of long MCA user identifier
D CDLML 1097 1100I 0
D* Length of long remote user identifier
D CDLRL 1101 1104I 0
D* Address of long MCA user identifier
D CDLMP 1105 1120*
D* Address of long remote user identifier
D CDLRP 1121 1136*
D* MCA security identifier
D CDMSI 1137 1176
D* Remote security identifier
D CDRSI 1177 1216
D* Ver:6 **
D* SSL CipherSpec
D CDSCS 1217 1248
D* Address of SSL peer name
D CDSPN 1249 1264*
D* Length of SSL peer name
D CDSPL 1265 1268I 0
D* Whether SSL client authentication is required
D CDSCA 1269 1272I 0
D* Keepalive interval
D CDKAI 1273 1276I 0
D* Local communications address
D CDLOA 1277 1324
D* Batch heartbeat interval
D CDBHB 1325 1328I 0
D* Ver:7 **
D* Header data compression list
D CDHCL0
D CDHCL1 1329 1332I 0
D CDHCL2 1333 1336I 0
D CDHCL 10I 0 DIM(2) OVERLAY(CDHCL0)
D* Message data compression list
D CDMCL0
D CDMCL1 1337 1340I 0
D CDMCL2 1341 1344I 0
D CDMCL3 1345 1348I 0
D CDMCL4 1349 1352I 0
D CDMCL5 1353 1356I 0

```

```

D CDMCL6          1357  1360I 0
D CDMCL7          1361  1364I 0
D CDMCL8          1365  1368I 0
D CDMCL9          1369  1372I 0
D CDMCL10         1373  1376I 0
D CDMCL11         1377  1380I 0
D CDMCL12         1381  1384I 0
D CDMCL13         1385  1388I 0
D CDMCL14         1389  1392I 0
D CDMCL15         1393  1396I 0
D CDMCL16         1397  1400I 0
D CDMCL           10I 0 DIM(16) OVERLAY(CDMCL0)
D* Channel rank
D CDCWCR          1401  1404I 0
D* Channel priority
D CDCWCP          1405  1408I 0
D* Channel weight
D CDCWCW          1409  1412I 0
D* Channel monitoring
D CDCHLMON        1413  1416I 0
D* Channel statistics
D CDCHLST         1417  1420I 0
D* Ver:8 **
D* Limit on sharing conversations
D CDSHC           1421  1424I 0
D* Message property control
D CDPRC           1425  1428I 0
D* Limit on SVRCONN channel instances
D CDMXIN          1429  1432I 0
D* Limit on SVRCONN channel instances per client
D CDMXIC          1433  1436I 0
D* Client channel weight
D CDCLNCHLW       1437  1440I 0
D* Connection affinity
D CDCONNAFF       1441  1444I 0
D* Ver:9 **
D* Batch data limit
D CDBDL           1445  1448I 0
D* Use Dead Letter Queue
D CDUDLQ          1449  1452I 0
D* Default client reconnect option
D CDDRCN          1453  1456I 0
D* Ver:10 **

```

System/390 汇编程序声明

此声明是 MQCD 结构的 System/390 汇编程序声明。

```

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_MAXMSGLLENGTH 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_NONPERSISTENTMSGSPD | DS | F | Speed at which nonpersistent messages are sent |
| * | | | |
| MQCD_STRUCLNGTH | 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_KEEPLIVEINTERVAL | 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 Affinity |

| | | | |
|---------------------|-----|-----------------|---------------------------------|
| MQCD_BATCHDATALIMIT | DS | F | Batch data limit |
| MQCD_USEDLO | 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) | |

Visual Basic 声明

此声明是 MQCD 结构的 Visual Basic 声明。

在 Visual Basic 中，MQCD 结构可以与 MQCONN 调用上的 MQCNO 结构配合使用。

| | | |
|-----------------------|---------------|--|
| 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 | | |

更改通道出口中的 MQCD 字段

通道出口可以更改 MQCD 中的字段。但是，通常不会对这些更改执行操作，除非在列出的情况下。

如果通道出口程序更改 MQCD 数据结构中的字段，那么 WebSphere MQ 通道进程通常会忽略新值。但是，新值仍保留在 MQCD 中并传递到出口链中的所有剩余出口，同时会传递到任何共享此通道实例的对话中。

如果 SharingConversations 在 MQCXP 结构中设置为 FALSE，那么可以根据出口程序类型，通道类型和出口原因码对某些字段进行更改。下表显示了可以更改并影响通道行为的字段，以及在何种情况下。如果出口程序在任何其他情况下更改其中一个字段，或者未列出任何字段，那么通道进程将忽略新值。新值将保留在 MQCD 中，并传递到出口链中的任何剩余出口以及共享通道实例的任何对话。

在调用任何类型的出口程序以进行初始化 (MQXR_INIT) 时，只要 MQCXP SharingConversations 设置为 FALSE，就可以更改任何类型的通道的 ChannelName 字段。无论 MQCXP SharingConversations 的值如何，只有安全出口才能更改 MCAUserIdentifier 字段。

| 字段 | 退出原因码 | 出口类型 | 通道类型 |
|---------------|-----------|------|---|
| ChannelName | MQXR_INIT | 全部 | 全部 |
| TransportType | MQXR_INIT | 全部 | 全部 |
| XmitQName | MQXR_INIT | 全部 | SDR 和 RCVR |
| ModeName | MQXR_INIT | 全部 | 全部 |
| TpName | MQXR_INIT | 全部 | 全部 |
| BatchSize | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |

| 字段 | 退出原因码 | 出口类型 | 通道类型 |
|--------------------|--|------|--|
| DiscInterval | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| ShortRetry 计数 | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| shortRetryInterval | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| LongRetry 计数 | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| longRetryInterval | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| SeqNumber 回绕 | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| MaxMsgLength | MQXR_INIT | 全部 | 全部 |
| PutAuthority | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| DataConversion | MQXR_INIT | 全部 | 全部 |
| MCAUserIdentifier | MQXR_INIT , MQXR_INIT_SEC , MQXR_SEC_MSG , MQXR_SEC_PARMS | 安全性 | RCVR , RQSTR , SVRCONN 和 CLUSRCVR |

| 字段 | 退出原因码 | 出口类型 | 通道类型 |
|-----------------------|--|------|--|
| ConnectionName | MQXR_INIT | 全部 | SDR , SVR , RQSTR , CLNTCONN , CLUSSDR 和 CLUSRCVR |
| MsgRetryUserData | MQXR_INIT | 全部 | RCVR , RQSTR 和 CLUSRCVR |
| MsgRetry 计数 | MQXR_INIT | 全部 | RCVR , RQSTR 和 CLUSRCVR |
| MsgRetry 时间间隔 | MQXR_INIT | 全部 | RCVR , RQSTR 和 CLUSRCVR |
| HeartbeatInterval | MQXR_INIT | 全部 | 全部 |
| BatchInterval | MQXR_INIT | 全部 | SDR , SVR , CLUSSDR 和 CLUSRCVR |
| NonPersistentMsgSpeed | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| MCASecurityId | MQXR_INIT , MQXR_INIT_SEC , MQXR_SEC_MSG , MQXR_SEC_PARMS | 安全性 | SDR , SVR , RCVR , RQSTR , SVRCONN , CLUSSDR 和 CLUSRCVR |
| SSLCipherSpec | MQXR_INIT | 全部 | 全部 |
| SSLPeerNamePtr | MQXR_INIT | 全部 | 全部 |
| SSLPeerName 长度 | MQXR_INIT | 全部 | 全部 |
| SSLClientAuth | MQXR_INIT | 全部 | SVR , RCVR , RQSTR , SVRCONN 和 CLUSRCVR |
| KeepAliveInterval | MQXR_INIT | 全部 | 全部 |

| 字段 | 退出原因码 | 出口类型 | 通道类型 |
|----------------------|-----------|------|--|
| LocalAddress | MQXR_INIT | 全部 | SDR , SVR , RQSTR , CLNTCONN , CLUSSDR 和 CLUSRCVR |
| BatchHeartbeat | MQXR_INIT | 全部 | SDR , SVR , CLUSSDR 和 CLUSRCVR |
| HdrComp 列表 | MQXR_INIT | 全部 | 全部 |
| MsgComp 列表 | MQXR_INIT | 全部 | 全部 |
| ChannelMonitoring | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , SVRCONN , CLUSSDR 和 CLUSRCVR |
| ChannelStatistics | MQXR_INIT | 全部 | SDR , SVR , RCVR , RQSTR , CLUSSDR 和 CLUSRCVR |
| SharingConversations | MQXR_INIT | 全部 | SVRCONN 和 CLNTCONN |
| PropertyControl | MQXR_INIT | 全部 | SDR , SVR , CLUSSDR 和 CLUSRCVR |

MQCXP-通道出口参数

MQCXP 结构传递到消息通道代理程序 (MCA) ， 客户机连接通道或服务器连接通道调用的每种出口类型。

请参阅 MQ_CHANNEL_EXIT。

当出口向通道返回控制权时，通道将忽略描述中描述为 "出口的输入" 的字段。将不会保留通道出口参数块中出口更改的任何输入字段用于下次调用。对输入/输出字段 (例如， *ExitUserArea* 字段) 所作的更改仅保留用于该出口实例的调用。此类更改不能用于在同一通道上定义的不同出口之间或不同通道上定义的不同出口之间传递数据。

相关参考

[第 959 页的『字段』](#)

本主题列出 MQCXP 结构中的所有字段并描述每个字段。

[第 968 页的『C 声明』](#)

此声明是 MQCXP 结构的 C 声明。

[第 968 页的『COBOL 声明』](#)

此声明是 MQCXP 结构的 COBOL 声明。

第 969 页的『RPG 声明 (ILE)』

此声明是 MQCXP 结构的 RPG 声明。

第 970 页的『System/390 汇编程序声明』

此声明是 MQCXP 结构的 System/390 汇编程序声明。

字段

本主题列出 MQCXP 结构中的所有字段并描述每个字段。

StrucId (MQCHAR4)

此字段指定结构标识。

该值必须为:

MQCXP_STRUC_ID

通道出口参数结构的标识。

对于 C 编程语言, 还定义了常量 MQCXP_STRUC_ID_ARRAY; 此常量具有与 MQCXP_STRUC_ID 相同的值, 但是字符数组而不是字符串。

这是出口的输入字段。

Version (MQLONG)

此字段指定结构版本号。

该值取决于环境:

MQCXP_VERSION_1

Version-1 通道出口参数结构。

MQCXP_VERSION_2

Version-2 通道出口参数结构。

此字段在以下环境中具有此值: HP Integrity NonStop Server。

MQCXP_VERSION_3

Version-3 通道出口参数结构。

此字段在以下环境中具有此值: 未在其他位置列出的 UNIX 系统。

MQCXP_VERSION_4

Version-4 通道出口参数结构。

MQCXP_VERSION_5

Version-5 通道出口参数结构。

MQCXP_VERSION_6

Version-6 通道出口参数结构。

MQCXP_VERSION_8

Version-8 通道出口参数结构。

此字段在以下环境中具有此值: z/OS, AIX, HP-UX, Linux, IBM i, Solaris 和 Windows。

仅在结构的最新版本中存在的字段在字段的描述中标识为此类字段。以下常量指定当前版本的版本号:

MQCXP_CURRENT_VERSION

当前版本的通道出口参数结构。

该值取决于环境。

注: 引入新版本的 MQCXP 结构时, 不会更改现有部件的布局。因此, 出口必须检查版本号是否等于或大于包含出口需要使用的字段的最低版本。

这是出口的输入字段。

ExitId (MQLONG)

此字段指定要调用的出口类型, 并在进入出口例程时进行设置。

可能的值如下所示:

MQXT_CHANNEL_SEC_EXIT

通道安全出口。

MQXT_CHANNEL_MSG_EXIT

通道消息出口。

MQXT_CHANNEL_SEND_EXIT

通道发送出口。

MQXT_CHANNEL_RCV_EXIT

通道接收出口。

MQXT_CHANNEL_MSG_RETRY_EXIT

通道消息-重试出口。

MQXT_CHANNEL_AUTO_DEF_EXIT

通道自动定义出口。

在 z/OS 上, 仅 MQCHT_CLUSSDR 和 MQCHT_CLUSRCVR 类型的通道支持此类型的出口。

这是出口的输入字段。

ExitReason (MQLONG)

此字段指定调用出口的原因, 并在进入出口例程时进行设置。

它不由自动定义出口使用。可能的值如下所示:

MQXR_INIT

退出初始化。

此值指示正在首次调用出口。它允许出口获取并初始化它需要的任何资源 (例如: 内存)。

MQXR_TERM

退出终止。

此值指示出口即将终止。出口应释放自初始化以来获取的任何资源 (例如: 内存)。

MQXR_MSG

处理消息。

此值指示正在调用出口以处理消息。此值仅适用于通道消息出口。

MQXR_XMIT

处理传输。

此值仅适用于通道发送和接收出口。

MQXR_SEC_MSG

接收到安全消息。

此值仅适用于通道安全出口。

MQXR_INIT_SEC

启动安全交换。

此值仅适用于通道安全出口。

在使用 MQXR_INIT 调用接收方的安全出口之后, 始终会立即使用此原因调用接收方的安全出口, 以使其有机会启动安全交换。如果它拒绝商机 (通过返回 MQXCC_OK 而不是 MQXCC_SEND_SEC_MSG 或 MQXCC_SEND_AND_REQUEST_SEC_MSG), 那么将使用 MQXR_INIT_SEC 调用发送方的安全出口。

如果接收方的安全出口确实启动了安全交换 (通过返回 MQXCC_SEND_SEC_MSG 或 MQXCC_SEND_AND_REQUEST_SEC_MSG), 那么从不使用 MQXR_INIT_SEC; 而是使用 MQXR_SEC_MSG 调用发送方的安全出口来处理接收方的消息。(在任一情况下, 都会首先使用 MQXR_INIT 进行调用。)

除非其中一个安全出口请求终止通道 (通过将 *ExitResponse* 设置为 MQXCC_SUPPRESS_FUNCTION 或 MQXCC_CLOSE_CHANNEL), 否则安全交换必须在启动交换的一侧完成。因此, 如果使用 MQXR_INIT_SEC 调用安全出口并启动交换, 那么下次调用该出口时将使用 MQXR_SEC_MSG。无论是

否存在要处理的出口的安全消息，都会发生此情况。如果合作伙伴返回 MQXCC_SEND_SEC_MSG 或 MQXCC_SEND_AND_REQUEST_SEC_MSG，但如果合作伙伴返回 MQXCC_OK 或合作伙伴没有安全出口，那么将显示安全消息。如果没有要处理的安全性消息，那么将在 *DataLength* 为零的情况下重新调用启动端的安全性出口。

MQXR_RETRY

重试消息。

此值仅适用于消息重试出口。

MQXR_AUTO_CLUSSDR

集群发送方通道的自动定义。

此值仅适用于通道自动定义出口。

MQXR_AUTO_RECEIVER

接收方通道的自动定义。

此值仅适用于通道自动定义出口。

MQXR_AUTO_SVRCONN

自动定义服务器连接通道。

此值仅适用于通道自动定义出口。

MQXR_AUTO_CLUSRCVR

集群接收方通道的自动定义。

此值仅适用于通道自动定义出口。

MQXR_SEC_PARMS

安全性参数

此值仅适用于安全出口，并指示正在将 MQCSP 结构传递到该出口。有关更多信息，请参阅第 296 页的『MQCSP-安全性参数』

注：

1. 如果为通道定义了多个出口，那么在初始化 MCA 时，将使用 MQXR_INIT 调用每个出口。此外，在终止 MCA 时，将使用 MQXR_TERM 来调用它们。
2. 对于通道自动定义出口，如果 *Version* 小于 MQCXP_VERSION_4，那么不会设置 *ExitReason*。在本例中隐含了值 MQXR_AUTO_SVRCONN。

这是出口的输入字段。

ExitResponse (MQLONG)

此字段指定来自出口的响应。

此字段由出口设置为与 MCA 通信。它必须是下列其中一个值：

MQXCC_OK

出口已成功完成。

- 对于通道安全出口，此值指示消息传输现在可以正常进行。
- 对于通道消息重试出口，此值指示 MCA 必须等待 MQCXP 中 *MsgRetryInterval* 字段中的出口返回的时间间隔，然后重试该消息。

ExitResponse2 字段可能包含其他信息。

MQXCC_SUPPRESS_FUNCTION

抑制函数。

- 对于通道安全出口，此值指示必须终止通道。
- 对于通道消息出口，此值指示消息不再向其目标前进。相反，MCA 会生成异常报告消息 (如果原始消息的发送方请求了异常报告消息)，并将包含在原始缓冲区中的消息放在死信队列上 (如果发送方指定了 MQRO_DEAD_LETTER_Q)，或者将其废弃 (如果发送方指定了 MQRO_DISCARD_MSG)。

对于持久消息，如果发送方指定了 `MQRO_DEAD_LETTER_Q`，但放入死信队列失败，或者没有死信队列，那么原始消息将保留在传输队列中，并且不会生成报告消息。如果无法成功生成报告消息，那么原始消息也会保留在传输队列上。

在死信队列上的消息开头的 `MQDLH` 结构中的 *Feedback* 字段指示将消息放入死信队列的原因；此反馈代码也在异常报告消息的消息描述符中使用（如果发送方请求了一个）。

- 对于通道消息重试出口，此值指示 MCA 不等待并重试消息；相反，MCA 会立即继续其正常故障处理（消息放置在死信队列上或废弃，如消息发送方所指定）。
- 对于通道自动定义出口，必须指定 `MQXCC_OK` 或 `MQXCC_SUPPRESS_FUNCTION`。如果未指定这两个值，那么缺省情况下将采用 `MQXCC_SUPPRESS_FUNCTION`，并且将放弃自动定义。

通道发送和接收出口不支持此响应。

MQXCC_SEND_SEC_MSG

发送安全消息。

此值只能由通道安全性出口设置。它指示出口提供了必须传输到合作伙伴的安全消息。

MQXCC_SEND_AND_REQUEST_SEC_MSG

发送需要应答的安全消息。

此值只能由通道安全性出口设置。它指示

- 出口提供了可以传输给合作伙伴的安全消息，并且
- 出口需要来自合作伙伴的响应。如果未收到响应，那么必须终止通道，因为出口尚未决定是否可以继续通信。

MQXCC_SUPPRESS_EXIT

禁止退出。

- 此值可以由除安全出口或自动定义出口以外的所有类型的通道出口设置。当使用 `MQXR_TERM` 的 *ExitReason* 再次调用该出口时，它将禁止该出口的任何进一步调用（就像其名称在通道定义中为空白一样），直到通道终止为止。
- 如果消息重试出口返回此值，那么后续消息的消息重试将由 *MsgRetryCount* 和 *MsgRetryInterval* 通道属性正常控制。对于当前消息，MCA 按 *MsgRetryInterval* 通道属性给定的时间间隔执行未完成的重试次数，但仅当原因码是 MCA 通常会重试的原因码时（请参阅第 922 页的『MQCD-通道定义』中描述的 *MsgRetryCount* 字段）。未完成重试次数是 *MsgRetryCount* 属性的值，减去出口针对当前消息返回 `MQXCC_OK` 的次数；如果此次数为负数，那么 MCA 不会对当前消息执行进一步的重试。

MQXCC_CLOSE_CHANNEL

关闭通道。

此值可以由除自动定义出口以外的任何类型的通道出口设置。

如果未启用共享对话，那么此值将关闭通道。

如果启用了共享对话，那么此值将结束对话。如果此对话是通道上的唯一对话，那么通道也会关闭。

此字段是来自出口的输入/输出字段。

ExitResponse2 (MQLONG)

此字段指定来自出口的辅助响应。

此字段在进入出口例程时设置为零。它可以由出口设置以向 WebSphere MQ 通道功能提供更多信息。它不由自动定义出口使用。

出口可以设置以下一个或多个值。如果需要多个值，那么将添加这些值。将记录无效的组合；允许其他组合。

MQXR2_PUT_WITH_DEF_ACTION

使用缺省操作进行放置。

此值由接收方的通道消息出口设置。它指示将使用 MCA 的缺省操作（即 MCA 的缺省用户标识）或消息的 `MQMD`（消息描述符）中的上下文 *UserIdentifier* 来放置消息。

该值为零，对应于调用出口时设置的初始值。提供常量是为了文档目的。

MQXR2_PUT_WITH_DEF_USERID

使用缺省用户标识进行放置。

此值只能由接收方的通道消息出口设置。它指示将使用 MCA 的缺省用户标识来放置消息。

MQXR2_PUT_WITH_MSG_USERID

与消息的用户标识一起放置。

此值只能由接收方的通道消息出口设置。它指示将消息与上下文 *UserIdentifier* 一起放入消息的 MQMD (消息描述符) 中 (这可能已由出口修改)。

应仅设置 MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERID 和 MQXR2_PUT_WITH_MSG_USERID 中的一个。

MQXR2_USE_AGENT_BUFFER

使用代理程序缓冲区。

此值指示要传递的任何数据都在 *AgentBuffer* 中，而不是 *ExitBufferAddr* 中。

该值为零，对应于调用出口时设置的初始值。提供常量是为了文档目的。

MQXR2_USE_EXIT_BUFFER

使用出口缓冲区。

此值指示要传递的任何数据都在 *ExitBufferAddr* 中，而不是 *AgentBuffer* 中。

仅应设置 MQXR2_USE_AGENT_BUFFER 中的一个。

MQXR2_DEFAULT_CONTINUATION

缺省延续。

链中的下一个出口的延续取决于调用的最后一个出口的响应:

- 如果返回 MQXCC_SUPPRESS_FUNCTION 或 MQXCC_CLOSE_CHANNEL，那么不会调用链中的其他出口。
- 否则，将调用链中的下一个出口。

MQXR2_CONTINUE_CHAIN

继续执行下一个出口。

MQXR2_SUPPRESS_CHAIN

跳过链中的剩余出口。

这是出口的输入/输出字段。

Feedback (MQLONG)

此字段指定反馈代码。

此字段在进入出口例程时设置为 MQFB_NONE。

如果通道消息出口将 *ExitResponse* 字段设置为 MQXCC_SUPPRESS_FUNCTION，那么 *Feedback* 字段指定用于标识将消息放入死信 (undelivered-message) 队列的原因的反馈代码，并且还用于发送异常报告 (如果已请求)。在此情况下，如果 *Feedback* 字段为 MQFB_NONE，那么将使用以下反馈代码:

MQFB_STOPPED_BY_MSG_EXIT

消息已由通道消息出口停止。

MCA 不使用按通道安全性，发送，接收和消息重试出口在此字段中返回的值。

如果 *ExitResponse* 为 MQXCC_OK，那么不会使用自动定义出口在此字段中返回的值，否则将用于事件消息中的 *AuxErrorDataInt1* 参数。

这是来自出口的输入/输出字段。

MaxSegment 长度 (MQLONG)

此字段指定可在单个传输中发送的最大长度 (以字节为单位)。

它不由自动定义出口使用。它与通道发送出口相关，因为此出口必须确保不会将传输段的大小增大到大于 *MaxSegmentLength* 的值。长度包括出口不得更改的初始 8 字节。该值在启动通道时在 WebSphere MQ 通道功能之间协商。请参阅 [编写通道出口程序](#)，以获取有关段长度的更多信息。

如果 *ExitReason* 为 MQXR_INIT，那么此字段中的值没有意义。

这是出口的输入字段。

ExitUser 区域 (MQBYTE16)

此字段指定出口用户区域-可供出口使用的字段。

在第一次调用出口 (将 *ExitReason* 设置为 MQXR_INIT) 之前，会将其初始化为二进制零，此后出口对该字段所作的任何更改都将在该出口的调用中保留。

定义了以下值:

MQXUA_NONE

无用户信息。

对于字段的长度，该值为二进制零。

对于 C 编程语言，还定义了常量 MQXUA_NONE_ARRAY; 此常量具有与 MQXUA_NONE 相同的值，但是字符数组而不是字符串。

此字段的长度由 MQ_EXIT_USER_AREA_LENGTH 指定。这是出口的输入/输出字段。

ExitData (MQCHAR32)

此字段指定出口数据。

在出口例程的入口上设置此字段，以获取 WebSphere MQ 通道函数从通道定义中获取的信息。如果没有此类信息可用，那么此字段全部为空白。

此字段的长度由 MQ_EXIT_DATA_LENGTH 指定。

这是出口的输入字段。

如果 *Version* 小于 MQCXP_VERSION_2，那么此结构中的以下字段不存在。

MsgRetry 计数 (MQLONG)

此字段指定重试消息的次数。

第一次对特定消息调用出口时，此字段的值为零 (尚未尝试任何重试)。每次后续调用该消息的出口时，该值将由 MCA 递增 1。

这是出口的输入字段。如果 *ExitReason* 为 MQXR_INIT，那么此字段中的值没有意义。如果 *Version* 小于 MQCXP_VERSION_2，那么此字段不存在。

MsgRetry 时间间隔 (MQLONG)

此字段指定重试放置操作的最小时间间隔 (以毫秒为单位)。

首次针对特定消息调用出口时，此字段包含 *MsgRetryInterval* 通道属性的值。该出口可以使该值保持不变，或者修改该值以指定不同的时间间隔 (以毫秒计)。如果出口在 *ExitResponse* 中返回 MQXCC_OK，那么 MCA 在重试 MQOPEN 或 MQPUT 操作之前至少等待此时间间隔。指定的时间间隔必须为零或更大。

针对该消息调用出口的第二次和后续时间，此字段包含上次调用该出口所返回的值。

如果 *MsgRetryInterval* 字段中返回的值小于 0 或大于 999 999 999，并且 *ExitResponse* 为 MQXCC_OK，那么 MCA 将忽略 MQCXP 中的 *MsgRetryInterval* 字段并等待 *MsgRetryInterval* 通道属性指定的时间间隔。

这是出口的输入/输出字段。如果 *ExitReason* 为 MQXR_INIT，那么此字段中的值没有意义。如果 *Version* 小于 MQCXP_VERSION_2，那么此字段不存在。

MsgRetry 原因 (MQLONG)

此字段指定上次尝试放入消息的原因码。

此字段是先前尝试放入消息的原因码; 它是 MQRC_* 值之一。

这是出口的输出字段。如果 *ExitReason* 为 MQXR_INIT，那么此字段中的值没有意义。如果 *Version* 小于 MQCXP_VERSION_2，那么此字段不存在。

如果 *Version* 小于 MQCXP_VERSION_3，那么此结构中的以下字段不存在。

HeaderLength (MQLONG)

此字段指定头信息的长度。

此字段仅与消息出口和消息重试出口相关。该值是消息数据开头的路由头结构的长度；这些是 MQXQH 结构，MQMDE (消息描述扩展头) 和 (对于分发列表消息) MQDH 结构以及 MQXQH 结构后面的 MQOR 和 MQPMR 记录数组。

消息出口可以检查此头信息，并在必要时对其进行修改，但出口返回的数据必须仍为正确的格式。例如，出口不得在发送端对头数据进行加密或压缩，即使接收端的消息出口进行了补偿性更改。

如果消息出口以更改其长度的方式修改头信息 (例如，通过将另一个目标添加到分发列表消息)，那么它必须在返回之前相应地更改 *HeaderLength* 的值。

这是出口的输出字段。如果 *ExitReason* 为 MQXR_INIT，那么此字段中的值没有意义。如果 *Version* 小于 MQCXP_VERSION_3，那么该字段不存在。

PartnerName (MQCHAR48)

此字段指定合作伙伴的名称。

合作伙伴的名称，如下所示：

- 对于 SVRCONN 通道，它是客户机上的已登录用户标识。
- 对于所有其他类型的通道，它是合作伙伴的队列管理器名称。

初始化出口时，此字段为空白，因为队列管理器直到进行初始协商之后才知道伙伴的名称。

这是出口的输出字段。如果 *Version* 小于 MQCXP_VERSION_3，那么该字段不存在。

FAPLevel (MQLONG)

协商的格式和协议级别。

这是出口的输出字段。仅应在 IBM 服务人员的指导下对此字段进行更改。如果 *Version* 小于 MQCXP_VERSION_3，那么该字段不存在。

CapabilityFlags (MQLONG)

此字段指定功能标志。

定义了以下内容：

MQCF_NONE

没有标志。

MQCF_DIST_LISTS

支持分发列表。

这是出口的输出字段。如果 *Version* 小于 MQCXP_VERSION_3，那么该字段不存在。

ExitNumber (MQLONG)

此字段指定出口的序号。

出口的序号，在 *ExitId* 中定义的类型内。例如，如果要调用的出口是定义的第三个消息出口，那么此字段包含值 3。如果出口类型是不能为其定义出口列表 (例如，安全出口) 的出口类型，那么此字段的值为 1。

这是出口的输出字段。如果 *Version* 小于 MQCXP_VERSION_3，那么该字段不存在。

如果 *Version* 小于 MQCXP_VERSION_5，那么此结构中的以下字段不存在。

ExitSpace (MQLONG)

此字段指定传输缓冲区中保留给出口使用的字节数。

此字段仅与发送出口相关。它指定 WebSphere MQ 通道功能在传输缓冲区中保留的空间量 (以字节计), 以供出口使用。此字段允许出口向传输缓冲区添加少量数据 (通常不超过数百字节), 以供另一端的补充接收出口使用。发送出口添加的数据必须由接收出口除去。

在 z/OS 上, 该值始终为零。

注: 此设施不得用于发送大量数据, 因为它可能会降低性能, 甚至禁止通道的操作。

通过设置 *ExitSpace*, 可以保证出口在传输缓冲区中始终至少有该数量的字节可供出口使用。但是, 如果传输缓冲区中有可用空间, 那么出口可以使用小于保留的量或大于保留的量。缓冲区中的出口空间是在现有数据之后提供的。

仅当 *ExitReason* 具有值 MQXR_INIT; 在所有其他情况下, 将忽略出口返回的值时, 出口才能设置 *ExitSpace*。对于出口的输入, *ExitSpace* 对于 MQXR_INIT 调用为零, 并且是 MQXR_INIT 调用在其他情况下返回的值。

如果 MQXR_INIT 调用返回的值为负数, 或者在为链中的所有发送出口保留所请求的出口空间后, 在消息数据的传输缓冲区中可用字节数少于 1024 字节, 那么 MCA 将输出错误消息并关闭通道。同样, 如果在数据传输期间, 发送出口链中的出口分配的用户空间多于保留的用户空间, 因此在消息数据的传输缓冲区中保留的字节数少于 1024 字节, 那么 MCA 将输出错误消息并关闭通道。1024 的限制允许通道的控制和管理流由发送出口链处理, 而不需要对流进行分段。

这是出口的输入/输出字段 (如果 *ExitReason* 是 MQXR_INIT) 和所有其他情况下的输入字段。如果 *Version* 小于 MQCXP_VERSION_5, 那么此字段不存在。

SSLCertUser 标识 (MQCHAR12)

此字段指定与远程证书关联的 *UserId*。

它在除 z/OS 以外的所有平台上都为空白

这是出口的输入字段。如果 *Version* 小于 MQCXP_VERSION_6, 那么该字段不存在。

SSLRemCertIssName 长度 (MQLONG)

此字段指定 *SSLRemCertRemoteIssuerNamePtr* 指向的远程证书颁发者的完整专有名称的长度 (以字节计)。

这是出口的输入字段。如果 *Version* 小于 MQCXP_VERSION_6, 那么该字段不存在。如果该值不是 SSL 通道, 那么该值为零。

SSLRemCertIssNamePtr (PMQVOID)

此字段指定远程证书签发者的完整专有名称的地址。

如果它不是 SSL 通道, 那么其值为空指针。

这是出口的输入字段。如果 *Version* 小于 MQCXP_VERSION_6, 那么该字段不存在。

注: 在 WebSphere MQ v7.1 发行版中, 通道安全出口在确定主题专有名称和颁发者专有名称时的行为已改变。有关更多信息, 请参阅 [通道安全出口程序](#)。

SecurityParms (PMQCSP)

此字段指定用于指定用户标识和密码的 MQSCP 结构的地址。

此字段的初始值为空指针。

这是出口的输入/输出字段。如果 *Version* 小于 MQCXP_VERSION_6, 那么该字段不存在。

CurHdr 压缩 (MQLONG)

此字段指定当前用于压缩头数据的方法。

它设置为下列其中一项:

MQCOMPRESS_NONE

不执行头数据压缩。

MQCOMPRESS_SYSTEM

执行头数据压缩。

该值可由发送通道的消息出口更改为从 MQCD 的 HdrCompList 字段访问的其中一个协商的受支持值。这将启用用于根据消息内容来压缩要为每条消息选择的头数据的方法。更改后的值仅用于当前消息。如果将属性更改为不受支持的值，那么通道结束。如果在发送通道的消息出口外部更改了该值，那么将忽略该值。

这是出口的输入/输出字段。如果 *Version* 小于 MQCXP_VERSION_6，那么该字段不存在。

CurMsg 压缩 (MQLONG)

此字段指定当前用于压缩消息数据的技术。

它设置为下列其中一项:

MQCOMPRESS_NONE

不执行头数据压缩。

MQCOMPRESS_RLE

使用运行长度编码执行消息数据压缩。

MQCOMPRESS_ZLIBFAST

使用 zlib 压缩技术来执行消息数据压缩。推荐使用快速压缩时间。

MQCOMPRESS_ZLIBHIGH

使用 zlib 压缩技术来执行消息数据压缩。推荐使用高级压缩。

该值可由发送通道的消息出口更改为从 MQCD 的 MsgCompList 字段访问的其中一个协商的受支持值。这使用于压缩要根据消息内容为每条消息决定的消息数据的方法成为可能。更改后的值仅用于当前消息。如果将属性更改为不受支持的值，那么通道结束。如果在发送通道的消息出口外部更改了该值，那么将忽略该值。

这是出口的输入/输出字段。如果 *Version* 小于 MQCXP_VERSION_6，那么该字段不存在。

Hconn (MQHCONN)

此字段指定出口在需要在出口中进行任何 MQI 调用时使用的连接句柄。

此字段与在客户机连接通道上运行的出口无关，其中包含值 MQHC_UNUSABLE_HCONN (-1)。

这是出口的输入字段。如果 *Version* 小于 MQCXP_VERSION_7，那么此字段不存在。

SharingConversations (MQBOOL)

此字段指定对话是当前唯一可以在此通道实例上运行的对话，还是当前可以在此通道实例上运行多个对话。

它还指示出口程序是否受制于 MQCD 被同时运行的另一个出口程序改变的风险。

此字段仅与在客户机连接或服务器连接通道上运行的出口程序相关。

它设置为下列其中一项:

FALSE

出口实例是当前可以在此通道实例上运行的唯一出口实例。这允许出口安全地更新 MQCD 字段，而不与其他通道实例上运行的其他出口发生争用。通道是否对 MQCD 字段进行更改由 [第 955 页的『更改通道出口中的 MQCD 字段』](#) 中的 MQCD 字段表定义。

TRUE

出口实例不是当前可以在此通道实例上运行的唯一出口实例。对 MQCD 进行的任何更改都不会由通道执行，但出于退出原因 (MQXR_INIT 除外)，[第 955 页的『更改通道出口中的 MQCD 字段』](#) 中的 MQCD 字段表中列出的更改除外。如果此出口更新 MQCD 字段，请通过在此通道实例上运行的出口之间提供序列化来确保没有来自同时在其他对话上运行的其他出口的争用。

这是出口的输入字段。如果 *Version* 小于 MQCXP_VERSION_7，那么此字段不存在。

MCAUserSource (MQLONG)

此字段指定提供的 MCA 用户标识的源。

它可以包含下列其中一个值:

MQUSRC_MAP

用户标识在 MCAUSER 属性中指定。

MQUSRC_CHANNEL

用户标识来自入站伙伴，或者在通道对象中定义的 MCAUSER 字段中指定。

这是出口的输出字段。如果版本低于 MQCXP_VERSION_8, 那么该字段不存在。

pEntry 点 (PMQIEP)

此字段指定 MQI 或 DCI 调用的接口入口点的地址。

如果 *Version* 小于 MQCXP_VERSION_8, 那么该字段不存在。

C 声明

此声明是 MQCXP 结构的 C 声明。

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
    MQCHAR4   StructId;           /* 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 声明

此声明是 MQCXP 结构的 COBOL 声明。

```
** 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.

```

RPG 声明 (ILE)

此声明是 MQCXP 结构的 RPG 声明。

```

D*..1....:....2.....3.....4.....5.....6.....7..
D* MQCXP Structure
D*
D* Structure identifier
D  CXSID          1      4
D* Structure version number
D  CXVER          5      8I 0
D* Type of exit
D  CXXID          9      12I 0
D* Reason for invoking exit
D  CXREA         13      16I 0
D* Response from exit
D  CXRES         17      20I 0
D* Secondary response from exit
D  CXRE2         21      24I 0
D* Feedback code
D  CXFB          25      28I 0
D* Maximum segment length
D  CXMSL         29      32I 0

```

```

D* Exit user area
D CXUA          33    48
D* Exit data
D CXDAT        49    80
D* Number of times the message has been retried
D CXMRC        81    84I 0
D* Minimum interval in milliseconds after which the put operation
D* should be retried
D CXMRI        85    88I 0
D* Reason code from previous attempt to put the message
D CXMRR        89    92I 0
D* Length of header information
D CXHDL        93    96I 0
D* Partner Name
D CXPNM        97    144
D* Negotiated Formats and Protocols level
D CXFAP        145   148I 0
D* Capability flags
D CXCAP        149   152I 0
D* Exit number
D CXEXN        153   156I 0
D* Number of bytes in transmission buffer reserved for exit to use
D CXHDL        157   160I 0
D* User identifier associated with remote SSL certificate
D CXSSLCU      161   172
D* Length of distinguished name of issuer of remote SSL certificate
D CXSRCINL     173   176I 0
D* Address of distinguished name of issuer of remote SSL certificate
D CXSRCINP     177   192*
D* Security parameters
D CXSECP       193   208*
D* Header data compression used for current message
D CXCHC        209   212I 0
D* Message data compression used for current message
D CXCMC        213   216I 0
D* Connection handle
D CXHCONN      217   220I 0
D* Multiple conversations possible on channel instance?
D CXSHARECONV  221   224I 0
D* Source of the provided MCA user ID
D MCAUSERSOURCE 225   228I 0

```

System/390 汇编程序声明

此声明是 MQCXP 结构的 System/390 汇编程序声明。

```

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_EXITSPEACE 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 |
| * MQCXP_CURMSGCOMPRESSION | DS | F | current message Message data compression used for |
| * MQCXP_HCONN | DS | F | current message Connection handle |
| MQCXP_SHARINGCONVERSATIONS | DS | F | Multiple conversations possible on |
| * MQCXP_MCAUSERSOURCE | DS | F | channel inst? Source of the provided MCA user ID |
| MQCXP_LENGTH | EQU | *-MQCXP | |
| | ORG | MQCXP | |
| MQCXP_AREA | DS | CL(MQCXP_LENGTH) | |

MQXWD-出口等待描述符

MQXWD 结构是 MQXWAIT 调用上的输入/输出参数。

此结构仅在 z/OS 上受支持。

相关参考

[第 971 页的『字段』](#)

本主题列出 MQXWD 结构中的所有字段并描述每个字段。

[第 972 页的『C 声明』](#)

此声明是 MQXWD 结构的 C 声明。

[第 972 页的『System/390 汇编程序声明』](#)

此声明是 MQXWD 结构的 System/390 汇编程序声明。

字段

本主题列出 MQXWD 结构中的所有字段并描述每个字段。

StrucId (MQCHAR4)

此字段指定结构标识。

该值必须为:

MQXWD_STRUC_ID

出口等待描述符结构的标识。

对于 C 编程语言，还定义了常量 MQXWD_STRUC_ID_ARRAY; 此常量具有与 MQXWD_STRUC_ID 相同的值，但是字符数组而不是字符串。

此字段的初始值为 MQXWD_STRUC_ID。

Version (MQLONG)

此字段指定结构版本号。

该值必须为:

MQXWD_VERSION_1

出口等待描述符结构的版本号。

此字段的初始值为 MQXWD_VERSION_1。

Reserved1 (MQLONG)

此字段是保留的。其值必须为零。

这是一个输入字段。

Reserved2 (MQLONG)

此字段是保留的。其值必须为零。

这是一个输入字段。

Reserved3 (MQLONG)

此字段是保留的。其值必须为零。

这是一个输入字段。

ECB (MQLONG)

此字段指定要等待的事件控制块。

此字段是要等待的事件控制块 (ECB)。在发出 MQXWAIT 调用之前，必须将其设置为零；成功完成时，它包含发布代码。

此字段是输入/输出字段。

C 声明

此声明是 MQXWD 结构的 C 声明。

```
typedef struct tagMQXWD MQXWD;
struct tagMQXWD {
    MQCHAR4  StrucId;    /* Structure identifier */
    MQLONG   Version;   /* Structure version number */
    MQLONG   Reserved1; /* Reserved */
    MQLONG   Reserved2; /* Reserved */
    MQLONG   Reserved3; /* Reserved */
    MQLONG   ECB;      /* Event control block to wait on */
};
```

System/390 汇编程序声明

此声明是 MQXWD 结构的 System/390 汇编程序声明。

```
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 出口参考

本部分主要提供编写 API 出口的程序员感兴趣的参考信息。

一般使用说明

注:

1. 所有出口函数都可以发出 MQXEP 调用；此调用专为从 API 出口函数使用而设计。
2. MQ_INIT_EXIT 函数无法发出除 MQXEP 以外的任何 MQ 调用。
3. 不能对当前连接发出 MQDISC 调用。
4. 如果出口函数发出 MQCONN 调用或带有 MQCNO_HANDLE_SHARE_NONE 选项的 MQCONN 调用，那么该调用将完成，原因码为 MQRC_ALREADY_CONNECTED，并且返回的句柄与作为参数传递到出口的句柄相同。
5. 通常，当 API 出口函数发出 MQI 调用时，不会以递归方式调用 API 出口。但是，如果出口函数使用 MQCNO_HANDLE_SHARE_BLOCK 或 MQCNO_HANDLE_SHARE_NO_BLOCK 选项发出 MQCONN 调用，那么该调用将返回新的共享句柄。这将为出口套件提供其自己的连接句柄，从而提供独立于应用程序的工作单元的工作单元。出口套件可以使用此句柄在其自己的工作单元中放入和获取消息，并落实或回退该工作单元；所有这些操作都可以在不以任何方式影响应用程序的工作单元的情况下完成。

由于出口函数正在使用与应用程序正在使用的句柄不同的连接句柄，因此出口函数发出的 MQ 调用会导致调用相关的 API 出口函数。因此，可以递归调用出口函数。请注意，MQAXP 中的 *ExitUserArea* 字段和出口链区域都具有连接句柄作用域。因此，出口函数不能使用这些区域以递归方式向自身的另一个实例发出其已处于活动状态的信号。

- 出口函数还可以在应用程序的工作单元中放置和获取消息。当应用程序落实或回退工作单元时，将落实或回退工作单元中的所有消息，而不考虑是谁将这些消息放在工作单元中(应用程序或出口功能)。但是，与其他情况相比，出口可能会导致应用程序超过系统限制(例如，超过工作单元中未落实的最大消息数)。

当出口函数以此方式使用应用程序的工作单元时，出口函数通常应避免发出 MQCMIT 调用，因为这将落实应用程序的工作单元，并且可能会损害应用程序的正确功能。但是，如果出口函数迂到阻止落实工作单元的严重错误(例如，将消息作为应用程序工作单元的一部分时发生错误)，那么出口函数有时可能需要发出 MQBACK 调用。调用 MQBACK 时，请注意确保应用程序工作单元边界不会更改。在此情况下，出口函数必须设置相应的值，以确保完成代码 MQCC_WARNING 和原因码 MQRC_BACKED_OUT 返回到应用程序，以便应用程序可以检测到工作单元已回退的事实。

如果出口函数使用应用程序的连接句柄来发出 MQ 调用，那么这些调用本身不会导致进一步调用 API 出口函数。

- 如果 MQXR_BEFORE 出口函数异常终止，那么队列管理器可能能够从故障中恢复。如果可以，那么队列管理器将继续处理，就像出口函数返回了 MQXCC_FAILED 一样。如果队列管理器无法恢复，那么应用程序将终止。
- 如果 MQXR_AFTER 出口函数异常终止，那么队列管理器可能能够从故障中恢复。如果可以，那么队列管理器将继续处理，就像出口函数返回了 MQXCC_FAILED 一样。如果队列管理器无法恢复，那么应用程序将终止。请注意，在后一种情况下，在工作单元外部检索的消息会丢失(这与从队列中除去消息后应用程序立即失败的情况相同)。
- MCA 进程执行两阶段落实。

如果 API 出口从预编译的 MCA 进程拦截 MQCMIT 并尝试在工作单元中执行操作，那么该操作将失败，原因码为 MQRC_UOW_NOT_AVAILABLE。

- 对于多安装环境，具有同时与 Websphere MQ V 7.0 和 V 7.1 配合使用的出口的唯一方法是在版本 7.0 与 mqm.Lib 链接的情况下编写出口，对于非主出口或重定位的出口，确保应用程序在应用程序启动之前找到当前与队列管理器关联的正确的 mqm.Lib。(例如，在启动应用程序之前运行 **setmqenv -m QM** 命令，即使队列管理器由 V 7.0 安装拥有也是如此。)
- 如果有多个 IBM WebSphere MQ 安装可用，请使用为较低版本的 IBM WebSphere MQ 编写的出口，因为在较高版本中添加的新功能可能无法用于较低版本。有关发行版之间的更改的更多信息，请参阅 [WebSphere MQ 7.5](#)。

IBM WebSphere MQ API 出口参数结构 (MQAXP)

MQAXP 结构(外部控制块)用作 API 出口的输入或输出参数。本主题还提供了有关队列管理器如何处理出口函数的信息。

MQAXP 具有以下 C 声明:

```
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 */
};
```

调用 API 出口中的函数时，将传递以下参数列表:

StrucId (MQCHAR4)-输入

出口参数结构标识, 值为:

```
MQAXP_STRUC_ID.
```

出口处理程序将入口上的此字段设置为每个出口函数。

版本 (MQLONG)-输入

结构版本号, 值为:

MQAXP_VERSION_1

V 1 API 出口参数结构。

MQAXP_VERSION_2

V 2 API 出口参数结构。

MQAXP_CURRENT_VERSION

API 出口参数结构的当前版本号。

出口处理程序将入口上的此字段设置为每个出口函数。

ExitId (MQLONG)-输入

出口标识, 在进入出口例程时设置, 指示出口类型:

MQXT_API_EXIT

API 出口。

ExitReason (MQLONG)-输入

调用出口的原因 (在进入每个出口函数时设置):

MQXR_CONNECTION

正在调用出口以在 MQCONN 或 MQCONNX 调用之前初始化自身, 或者在 MQDISC 调用之后结束自身。

MQXR_BEFORE

正在执行 API 调用之前或在 MQGET 上转换数据之前调用出口。

MQXR_AFTER

正在执行 API 调用后调用出口。

ExitResponse (MQLONG)-输出

来自出口的响应, 在进入每个出口函数时初始化为:

MQXCC_OK

正常继续。

此字段必须由出口函数设置, 以便将执行出口函数的结果传达给队列管理器。值必须为以下其中一项:

MQXCC_OK

出口功能已成功完成。正常继续。

此值可由所有 MQXR_* 出口函数设置。ExitResponse2 用于决定是否稍后在链中调用出口函数。

MQXCC_FAILED

由于发生错误, 退出函数失败。

此值可由所有 MQXR_* 出口函数设置。队列管理器将 CompCode 设置为 MQCC_FAILED, 并将原因设置为:

- MQRC_API_EXIT_INIT_ERROR (如果函数为 MQ_INIT_EXIT)
- MQRC_API_EXIT_TERM_ERROR (如果函数为 MQ_TERM_EXIT)
- 所有其他出口函数的 MQRC_API_EXIT_ERROR

该值集可以由链中稍后的出口函数进行更改。

ExitResponse2 被忽略; 队列管理器继续处理, 就像返回了 MQXR2_SUPPRESS_CHAIN 一样。

MQXCC_SUPPRESS_FUNCTION

禁止 WebSphere MQ API 函数。

此值只能由 MQXR_BEFORE 出口函数设置。它会绕过 API 调用。如果由 MQ_DATA_CONV_ON_GET_EXIT 返回，那么将绕过数据转换。队列管理器将 CompCode 设置为 MQCC_FAILED，将 "原因" 设置为 MQRC_SUPPRESSED_BY_EXIT，但该值集可以由链中稍后的出口函数进行更改。当出口留下这些参数时，将保留该调用的其他参数。ExitResponse2 用于决定是否稍后在链中调用出口函数。

如果此值由 MQXR_AFTER 或 MQXR_CONNECTION 出口函数设置，那么队列管理器将继续处理，就像已返回 MQXCC_FAILED 一样。

MQXCC_SKIP_FUNCTION

跳过 WebSphere MQ API 函数。

此值只能由 MQXR_BEFORE 出口函数设置。它会绕过 API 调用。如果由 MQ_DATA_CONV_ON_GET_EXIT 返回，那么将绕过数据转换。出口函数必须将 CompCode 和 "原因" 设置为要返回到应用程序的值，但该设置的值可以由链中稍后的出口函数进行更改。当出口留下这些参数时，将保留该调用的其他参数。ExitResponse2 用于决定是否稍后在链中调用出口函数。

如果此值由 MQXR_AFTER 或 MQXR_CONNECTION 出口函数设置，那么队列管理器将继续处理，就像已返回 MQXCC_FAILED 一样。

MQXCC_SUPPRESS_EXIT

禁止属于出口集的所有出口函数。

此值只能由 MQXR_BEFORE 和 MQXR_AFTER 出口函数设置。它会绕过属于此逻辑连接的这组出口的出口函数的所有后续调用。当使用 ExitReason MQXR_CONNECTION 调用 MQ_TERM_EXIT 函数时，此绕过将一直持续到发生逻辑断开连接请求。

出口函数必须将 CompCode 和 "原因" 设置为要返回到应用程序的值，但该设置的值可以由链中稍后的出口函数进行更改。当出口留下这些参数时，将保留该调用的其他参数。将忽略 ExitResponse2。

如果此值由 MQXR_CONNECTION 出口函数设置，那么队列管理器将继续处理，就像已返回 MQXCC_FAILED 一样。

有关 ExitResponse 与 ExitResponse2 之间的交互及其对出口处理的影响的信息，请参阅 [第 977 页的『队列管理器如何处理出口函数』](#)。

ExitResponse2 (MQLONG)-输出

这是辅助出口响应代码，用于限定 MQXR_BEFORE 出口函数的主出口响应代码。它已初始化为：

```
MQXR2_DEFAULT_CONTINUATION
```

在进入 WebSphere MQ API 调用出口函数时。然后，可以将其设置为下列其中一个值：

MQXR2_DEFAULT_CONTINUATION

是否继续链中的下一个出口，具体取决于 ExitResponse 的值。

如果 ExitResponse 是 MQXCC_SUPPRESS_FUNCTION 或 MQXCC_SKIP_FUNCTION，请稍后绕过 MQXR_BEFORE 链中的出口函数和 MQXR_AFTER 链中的匹配出口函数。调用 MQXR_AFTER 链中与 MQXR_BEFORE 链中较早的出口函数匹配的出口函数。

否则，调用链中的下一个出口。

MQXR2_SUPPRESS_CHAIN

抑制链。

对于此 API 调用，绕过 MQXR_BEFORE 链中稍后的出口函数以及 MQXR_AFTER 链中的匹配出口函数。调用 MQXR_AFTER 链中与 MQXR_BEFORE 链中较早的出口函数匹配的出口函数。

MQXR2_CONTINUE_CHAIN

继续链中的下一个出口。

有关 ExitResponse 与 ExitResponse2 之间的交互及其对出口处理的影响的信息，请参阅 [第 977 页的『队列管理器如何处理出口函数』](#)。

反馈 (MQLONG)-输入/输出

在退出函数调用之间传递反馈代码。此操作初始化为：

MQFB_NONE (0)

在调用链中的第一个出口的函数之前。

出口可以将此字段设置为任何值，包括任何有效的 MQFB_* 或 MQRC_* 值。出口还可以将此字段设置为 MQFB_APPL_FIRST 到 MQFB_APPL_LAST 范围内的用户定义的反馈值。

APICallerType (MQLONG)-输入

API 调用者类型，指示 WebSphere MQ API 调用者是队列管理器的外部还是内部：XX_ENCODE_CASE_ONE mqxact_external 或 MQXACT_INTERNAL。

ExitUser 区域 (MQBYTE16)-输入/输出

用户区域，可用于与特定 ExitInfo 对象关联的所有出口。在对 hconn 调用第一个出口函数 (MQ_INIT_EXIT) 之前，将其初始化为 MQXUA_NONE (ExitUser 区域的长度为二进制零)。从那时起，将在同一出口的函数调用中保留出口函数对此字段所作的任何更改。

此字段与 4 MQLONG 的倍数对齐。

出口还可以锚定他们从此区域分配的任何存储器。

对于每个 hconn，出口链中的每个出口都具有不同的 ExitUser 区域。ExitUser 区域不能由链中的出口共享，并且一个出口的 ExitUser 区域的内容不可用于链中的另一个出口。

对于 C 程序，常量 MQXUA_NONE_ARRAY 也使用与 MQXUA_NONE 相同的值定义，但定义为字符数组而不是字符串。

此字段的长度由 MQ_EXIT_USER_AREA_LENGTH 指定。

ExitData (MQCHAR32)-输入

出口数据，在每个出口函数的输入上设置为出口中提供的特定于出口的数据的 32 个字符。如果在出口中未定义任何值，那么此字段全部为空白。

此字段的长度由 MQ_EXIT_DATA_LENGTH 指定。

ExitInfo 名称 (MQCHAR48)-输入

出口信息名称，在对节中的出口定义中指定的 ApiExit_name 的每个出口函数的输入时设置。

ExitPDArea (MQBYTE48)-输入/输出

对于出口函数的每次调用，初始化为 MQXPDA_NONE (字段长度为二进制零) 的问题确定区域。

对于 C 程序，常量 MQXPDA_NONE_ARRAY 也定义为具有与 MQXPDA_NONE 相同的值，但定义为字符数组而不是字符串。

出口处理程序始终在出口结束时将此区域写入 WebSphere MQ 跟踪，即使该功能成功也如此。

此字段的长度由 MQ_EXIT_PD_AREA_LENGTH 提供。

QMgrName (MQCHAR48)-输入

由于处理 WebSphere MQ API 调用而调用出口的应用程序所连接到的队列管理器的名称。

如果 MQCONN 或 MQCONNX 调用上提供的队列管理器的名称为空，那么无论应用程序是服务器还是客户机，此字段仍设置为应用程序所连接的队列管理器的名称。

出口处理程序将入口上的此字段设置为每个出口函数。

此字段的长度由 MQ_Q_MGR_NAME_LENGTH 提供。

ExitChainAreaPtr (PMQACH)-输入/输出

这用于在链中不同出口的调用之间传递数据。在调用出口链中第一个出口的函数 (带有 ExitReason MQXR_CONNECTION 的 MQ_INIT_EXIT) 之前，将其设置为 NULL 指针。出口在一次调用时返回的值将传递到下一次调用。

有关如何使用出口链区域的更多详细信息，请参阅第 980 页的『[出口链区域和出口链区域头 \(MQACH\)](#)』。

Hconfig (MQHCONFIG)-输入

配置句柄，表示正在初始化的函数集。此值由 MQ_INIT_EXIT 函数上的队列管理器生成，稍后将传递到 API 出口函数。它设置在每个出口函数的入口上。

您可以使用 Hconfig 作为 MQIEP 结构的指针来执行 MQI 和 DCI 调用。在将 HConfig 参数用作 MQIEP 结构的指针之前，必须检查 HConfig 的前 4 个字节是否与 MQIEP 结构的 StrucId 匹配。

函数 (MQLONG)-输入

函数标识，有效值为第 981 页的『外部常量』中描述的 MQXF_* 常量。

根据导致调用出口的 WebSphere MQ API 调用，出口处理程序在进入每个出口函数时将此字段设置为正确的值。

ExitMsgHandle (MQHMSG)-输入/输出

当函数为 MQXF_GET 且 ExitReason 为 MQXR_AFTER 时，将在此字段中返回有效消息句柄，允许 API 出口访问消息描述符字段以及在注册 API 出口时与 MQXEPO 结构中指定的 ExitProperties 字符串相匹配的任何其他属性。

如果指定了 MQGMO 结构中的 MsgHandle，那么在 ExitMsgHandle 中返回的任何非消息描述符属性都将不可用，或者在消息数据中不可用。

当函数为 MQXF_GET 且 ExitReason 为 MQXR_BEFORE 时，如果出口程序将此字段设置为 MQHM_NONE，那么它将禁止填充 ExitMsgHandle 属性。

如果版本低于 MQAXP_VERSION_2，那么不会设置此字段。

队列管理器如何处理出口函数

队列管理器在从出口函数返回时执行的处理取决于 ExitResponse 和 ExitResponse2。

第 977 页的表 593 汇总了 MQXR_BEFORE 出口函数的可能组合及其影响，显示：

- 设置 API 调用的 CompCode 和 Reason 参数的人员
- 是否调用 MQXR_BEFORE 链中的其余出口函数以及 MQXR_AFTER 链中的匹配出口函数
- 是否调用 API 调用

对于 MQXR_AFTER 出口函数：

- 以与 MQXR_BEFORE 相同的方式设置 CompCode 和 Reason
- 将忽略 ExitResponse2 (将始终调用 MQXR_AFTER 链中的其余出口函数)
- MQXCC_SUPPRESS_FUNCTION 和 MQXCC_SKIP_FUNCTION 无效

对于 MQXR_CONNECTION 出口函数：

- 以与 MQXR_BEFORE 相同的方式设置 CompCode 和 Reason
- ExitResponse2 被忽略
- MQXCC_SUPPRESS_FUNCTION，MQXCC_SKIP_FUNCTION 和 MQXCC_SUPPRESS_EXIT 无效

在所有情况下，如果出口或队列管理器设置了 CompCode 和 Reason，那么可以通过稍后调用的出口或通过 API 调用 (如果稍后调用 API 调用) 来更改值集。

| ExitResponse 的值 | CompCode 和原因集 | ExitResponse2 的值 (缺省延续) 链 | ExitResponse2 (缺省延续) API 的值 |
|-------------------------|---------------|---------------------------|-----------------------------|
| MQXCC_OK | 退出 | Y | Y |
| MQXCC_SUPPRESS_EXIT | 退出 | Y | Y |
| MQXCC_SUPPRESS_FUNCTION | 队列管理器 | N | N |
| MQXCC_SKIP 函数 | 退出 | N | N |
| MQXCC_FAILED | 队列管理器 | N | N |

客户机如何处理出口函数

通常，客户机处理出口函数的方式与服务器应用程序处理出口函数的方式相同，此结构中的 *QMgrName* 属性适用 (无论该函数是在服务器上还是在客户机上)。

但是，客户机没有 *mqs.ini* 文件的概念，因此 *ApiExitCommon* 和 *APIExitTemplate* 节不适用。只有 *ApiExitLocal* 节适用，此节在 *mqclient.ini* 文件中配置。

IBM WebSphere MQ API 出口上下文结构 (MQAXC)

MQAXC 结构 (外部控制块) 用作 API 出口的输入参数。

MQAXC 具有以下 C 声明:

```
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 */
};
```

MQAXC 的参数为:

StrucId (MQCHAR4)-输入

出口上下文结构标识，值为 MQAXC_STRUC_ID。对于 C 程序，还定义了常量 MQAXC_STRUC_ID_ARRAY，其值与 MQAXC_STRUC_ID 相同，但作为字符数组而不是字符串。

出口处理程序将入口上的此字段设置为每个出口函数。

版本 (MQLONG)-输入

结构版本号，值为:

MQAXC_VERSION_2

出口上下文结构的版本号。

MQAXC_CURRENT_VERSION

出口上下文结构的当前版本号。

出口处理程序将入口上的此字段设置为每个出口函数。

环境 (MQLONG)-输入

发出 WebSphere MQ API 调用的环境，该调用导致驱动出口函数。此字段的有效值为:

MQXE_OTHER

此值与 API 出口在从服务器应用程序调用该出口时看到的调用一致。这意味着 API 出口不会在客户机上运行，也不会看到任何不同的情况。

如果出口确实需要确定它是否在客户机上运行，那么出口可以通过查看 *ChannelName* 和 *ChannelDefinition* 字段来执行此操作。

MQXE_MCA

消息通道代理程序

MQXE_MCA_SVRCONN

代表客户机执行操作的消息通道代理程序

MQXE_COMMAND_SERVER

命令服务器

MQXE_MQSC

runmqsc 命令解释器

出口处理程序将入口上的此字段设置为每个出口函数。

UserId (MQCHAR12)-输入

与应用程序关联的用户标识。特别是，对于客户机连接，此字段包含采用的用户的用户标识，而不是运行通道代码的用户标识。如果空白用户标识来自客户机，那么不会对已在使用的用户标识进行任何更改。即，不采用新的用户标识。

出口处理程序将入口上的此字段设置为每个出口函数。此字段的长度由 MQ_USER_ID_LENGTH 给出。

对于客户机，这是从客户机发送到服务器的用户标识。请注意，这可能不是客户机在队列管理器中针对其运行的有效用户标识，因为可能存在更改用户标识的 MCAUser 或 CHLAUTH 配置。

SecurityId (MQBYTE40)-输入

对运行应用程序的用户标识的扩展。其长度由 MQ_SECURITY_ID_LENGTH 提供。

对于客户机，这是从客户机发送到服务器的用户标识。请注意，这可能不是客户机在队列管理器中针对其运行的有效用户标识，因为可能存在更改用户标识的 MCAUser 或 CHLAUTH 配置。

ConnectionName (MQCHAR264)-输入

连接名称字段，设置为客户机的地址。例如，对于 TCP/IP，它将是客户机 IP 地址。

此字段的长度由 MQ_CONN_NAME_LENGTH 给出。

对于客户机，这是队列管理器的伙伴地址。

LongMCAUserIdLength (MQLONG)-输入

长 MCA 用户标识的长度。

当 MCA 连接到队列管理器时，此字段设置为长 MCA 用户标识的长度 (如果没有这样的标识，则设置为零)。

对于客户机，这是客户机长用户标识。

LongRemoteUserId 长度 (MQLONG)-输入

长远程用户标识的长度。

当 MCA 连接到队列管理器时，此字段设置为长远程用户标识的长度。否则，此字段将设置为零

对于客户机，请将此字段设置为零。

LongMCAUserIdPtr (MQPTR)-输入

长 MCA 用户标识的地址。

当 MCA 连接到队列管理器时，此字段设置为长 MCA 用户标识的地址 (如果没有这样的标识，那么设置为空指针)。

对于客户机，这是客户机长用户标识。

LongRemoteUserIdPtr (MQPTR)-输入

长远程用户标识的地址。

当 MCA 连接到队列管理器时，此字段设置为长远程用户标识的地址 (如果没有此类标识，那么设置为空指针)。

对于客户机，请将此字段设置为零。

ApplName (MQCHAR28)-输入

发出 WebSphere MQ API 调用的应用程序或组件的名称。

生成 ApplName 的规则与生成 MQPUT 的缺省名称的规则相同。

通过查询操作系统以获取程序名，可找到此字段的值。其长度由 MQ_APPL_NAME_LENGTH 指定。

ApplType (MQLONG)-输入

发出 WebSphere MQ API 调用的应用程序或组件的类型。

对于编译应用程序的平台，该值为 MQAT_DEFAULT，或者等于其中一个定义的 MQAT_* 值。
出口处理程序将入口上的此字段设置为每个出口函数。

ProcessId (MQPID)-输入

操作系统进程标识。

在适用的情况下，出口处理程序会在进入每个出口函数时设置此字段。

ThreadId (MQTID)-输入

MQ 线程标识。这是 MQ 跟踪和 FFST 转储中使用的相同标识，但可能与操作系统线程标识不同。

在适用的情况下，出口处理程序会在进入每个出口函数时设置此字段。

ChannelName (MQCHAR)-输入

用空白填充的通道名称 (如果适用) 和已知的通道名称。

如果不适用，那么此字段将设置为 NULL 字符。

Reserved1 (MQBYTE4)-输入

此字段是保留的。

ChanneDefinition (PMQCD)-输入

指向正在使用的通道定义 (如果适用且已知) 的指针。

如果不适用，那么此字段将设置为 NULL 字符。

请注意，仅当连接正在代表 WebSphere MQ 通道进行处理并且已读取该通道定义时，才会完成该指针。

尤其是，当对通道进行第一次 MQCONN 调用时，不会在服务器上提供通道定义。此外，如果指针被填充，指针指向的结构 (和任何子结构) 必须被视为只读；结构的任何更新都将导致不可预测的结果，并且不受支持。

对于客户机，除具有为客户机指定的值的字段以外的字段包含适用于客户机应用程序的值。

出口链区域和出口链区域头 (MQACH)

如果需要，出口函数可以获取出口链区域的存储器，并将 MQAXP 中的 ExitChainAreaPtr 设置为指向此存储器。

出口 (相同或不同的出口功能) 可以获取多个出口链区域并将它们链接在一起。从出口处理程序调用时，必须仅从此列表中添加或删除出口链区域。这可确保不存在由不同线程同时添加或删除列表中的区域所导致的序列化问题。

出口链区域必须以 MQACH 头结构开头，其 C 声明为：

```
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 */
};
```

出口链区域头中的字段为：

StrucId (MQCHAR4)-输入

出口链区域结构标识，具有 MQACH_STRUC_ID 的初始值 (由 MQACH_DEFAULT 定义)。

对于 C 程序，还定义了常量 MQACH_STRUC_ID_ARRAY；它具有与 MQACH_STRUC_ID 相同的值，但作为字符数组而不是字符串。

版本 (MQLONG)-输入

结构版本号，如下所示：

MQACH_VERSION_1

出口参数结构的版本号。

MQACH_CURRENT_VERSION

出口上下文结构的当前版本号。

MQACH_DEFAULT 定义的此字段的初始值为 MQACH_CURRENT_VERSION。

注: 如果引入此结构的新版本, 那么现有部件的布局不会更改。出口函数必须检查版本号是否等于或大于包含出口函数需要使用的字段的最低版本。

StrucLength (MQLONG)-输入

MQACH 结构的长度。出口可以使用此字段来确定出口数据的开始, 并将其设置为出口创建的结构长度。

MQACH_DEFAULT 定义的此字段的初始值为 MQACH_CURRENT_LENGTH。

ChainArea 长度 (MQLONG)-输入

出口链区域长度, 设置为当前出口链区域的整体长度, 包括 MQACH 头。

此字段的初始值 (由 MQACH_DEFAULT 定义) 为零。

ExitInfo 名称 (MQCHAR48)-输入

出口信息名称。

当出口创建 MQACH 结构时, 它必须使用自己的 ExitInfo 名称来初始化此字段, 以便以后此 MQACH 结构可以由此出口的另一个实例或协作出口找到。

此字段的初始值 (由 MQACH_DEFAULT 定义) 是长度为零的字符串 ({""}).

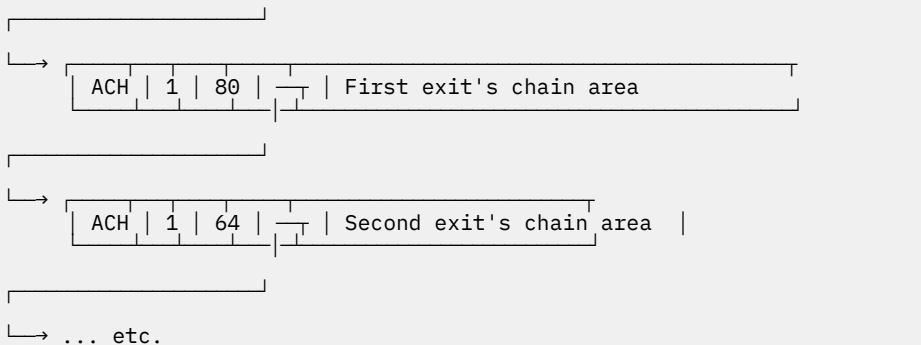
NextChainAreaPtr (PMQACH)-输入

指向下一个出口链区域的指针, 其初始值由 MQACH_DEFAULT 定义, 为空指针 (NULL)。

出口函数必须释放它们获取的任何出口链区域的存储器, 并处理链指针以从列表中除去它们的出口链区域。

出口链区域可按如下所示构造:

MQAXP.ExitChainAreaPtr —┐



外部常量

使用本主题作为可用于 API 的外部常量的参考信息。

以下外部常量可用于 API 出口:

MQXF_* (出口函数标识)

| | | |
|-----------------------|----|-------------|
| 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_* (退出原因)

| | | |
|-----------------|---|-------------|
| MQXR_BEFORE | 1 | X'00000001' |
| MQXR_AFTER | 2 | X'00000002' |
| MQXR_CONNECTION | 3 | X'00000003' |

MQXE_* (环境)

| | | |
|---------------------|---|-------------|
| 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*_* (其他常量)

| | | |
|-----------------------|--|--|
| 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*_* (空常量)

| | |
|-------------------|-------------------------|
| MQXPDA_NONE | X'00...00' (48 nulls) |
| MQXPDA_NONE_ARRAY | '\0','\0',...,'\0','\0' |

MQXCC_* (完成代码)

| | |
|--------------|----|
| MQXCC_FAILED | -8 |
|--------------|----|

MQRC_* (原因码)

MQRC_API_EXIT_ERROR 2374 X'00000946'

出口函数调用返回了无效的响应代码，或者以某种方式失败，并且队列管理器无法确定要执行的下一个操作。

检查 MQAXP 的 ExitResponse 和 ExitResponse2 字段以确定错误响应代码，并更改出口以返回有效响应代码。

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'

队列管理器在初始化 API 出口函数的执行环境时迁到错误。

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'

队列管理器在关闭 API 出口函数的执行环境时迁到错误。

MQRC_EXIT_REASON_ERROR 2377 X'00000949'

在出口入口点注册调用 (MQXEP) 调用上提供的 ExitReason 字段的值出错。

检查 ExitReason 字段的值以确定并更正错误的退出原因值。

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'

"保留" 字段的值出错。

检查 "保留" 字段的值以确定并更正 "保留" 值。

C 语言类型定义

本主题提供有关与以 C 语言提供的 API 出口关联的 typedefs 的信息。

以下是与 API 出口关联的 C 语言类型定义：

```
typedef PMLONG      MQPOINTER PPMQLONG;
typedef PMQBYTE     MQPOINTER PPMQBYTE;
typedef PMQHOBJS   MQPOINTER PPMQHOBJS;
typedef PMQOD       MQPOINTER PPMQOD;
typedef PMQMD       MQPOINTER PPMQMD;
typedef PMQPMO      MQPOINTER PPMQPMO;
typedef PMQGM       MQPOINTER PPMQGM;
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;
```

出口入口点注册调用 (MQXEP)

使用此信息来了解 MQXEP，MQXEP C 语言调用和 MQXEP C 函数原型。

使用 MQXEP 调用来执行以下操作：

1. 注册要在其中调用出口函数的前后 WebSphere MQ API 出口调用点
2. 指定出口函数入口点
3. 注销出口函数入口点

您通常会在 MQ_INIT_EXIT 出口函数中对 MQXEP 调用进行编码，但在任何后续出口函数中指定这些调用。

如果使用 MQXEP 调用来注册已注册的出口函数，那么第二个 MQXEP 调用将成功完成，从而替换已注册的出口函数。

如果使用 MQXEP 调用来注册 NULL 出口函数，那么 MQXEP 调用将成功完成并注销该出口函数。

如果 MQXEP 调用用于在连接请求的生命周期内注册，注销和重新注册特定出口函数，那么将重新激活先前注册的出口函数。仍分配并与此出口函数实例关联的任何存储器都可供出口函数使用。(此存储器通常在调用终止出口函数期间释放。)

MQXEP 的接口为:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

其中:

Hconfig (MQHCONFIG)-输入

配置句柄，表示包含正在初始化的函数集的 API 出口。此值由队列管理器在调用 MQ_INIT_EXIT 函数之前立即生成，并在 MQAXP 中传递到每个 API 出口函数。

ExitReason (MQLONG)-输入

正在注册入口点的原因，来自以下原因:

- 连接级别初始化或终止 (MQXR_CONNECTION)
- 在 WebSphere MQ API 调用 (MQXR_BEFORE) 之前
- 在 WebSphere MQ API 调用之后 (MQXR_AFTER)

函数 (MQLONG)-输入

函数标识，有效值为 MQXF_ * 常量 (请参阅第 981 页的『外部常量』)。

EntryPoint (PMQFUNC)-输入

要注册的出口函数的入口点的地址。值 NULL 指示未提供出口函数，或者正在注销出口函数的先前注册。

ExitOpts(MQXEPO)

API 出口可以指定用于控制 API 出口注册方式的选项。如果为此字段指定了空指针，那么将采用 MQXEPO 结构的缺省值。

CompCode (MQLONG)-输出

完成代码，有效值为:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因 (MQLONG)-输出

限定完成代码的原因码。

如果完成代码为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码为 MQCC_FAILED:

MQRC_HCONFIG_ERROR

(2280, X'8E8') 提供的配置句柄无效。使用 MQAXP 中的配置句柄。

MQRC_EXIT_REASON_ERROR

(2377, X'949') 提供的出口函数调用原因无效或对提供的出口函数标识无效。

请使用其中一个有效的出口函数调用原因 (MQXR_ * 值)，或者使用有效的函数标识和出口原因组合。(请参阅第 985 页的表 594。)

MQRC_FUNCTION_ERROR

(2281, X'8E9') 由于 API 出口原因，提供的函数标识无效。下表显示了函数标识和 ExitReasons 的有效组合。

| 表 594: 函数标识和 <i>ExitReasons</i> 的有效组合 | |
|---|---------------------------|
| 函数 | ExitReason |
| MQXF_INIT MQXF_TERM | MQXR_CONNECTION |
| 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') 由于资源问题, 尝试注册或注销出口函数失败。

MQRC_UNEXPECTED_ERROR

(2195, 'X'893') 尝试注册或注销出口函数意外失败。

MQRC_PROPERTY_NAME_ERROR

(2442, 'X'098A') ExitProperties 名称无效。

MQRC_XEPO_ERROR

(2507, 'X'09CB') 出口选项结构无效。

MQXEP C 语言调用

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

参数列表的声明:

```
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 C 函数原型

```
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 */

```

出口函数

本部分提供了一些一般信息来帮助您使用函数调用，并描述了如何调用各个出口函数。

使用此信息来了解 API 出口例程的一般规则，以及设置和清除出口执行环境。

API 出口例程的一般规则

调用 API 出口例程时，以下一般规则适用。

- 在所有情况下，在验证 API 调用参数之前以及在任何安全性检查 (对于 MQCONN, MQCONNX 或 MQOPEN) 之前，都会驱动 API 出口函数。
- 在出口例程中输入和输出的字段值为：
 - 在前 WebSphere MQ API 出口函数的输入中，字段的值可以由应用程序设置，也可以由先前的出口函数调用设置。
 - 在前 WebSphere MQ API 出口函数的输出中，字段的值可以保持不变，也可以由出口函数设置为其他值。
 - 在之后 WebSphere MQ API 出口函数的输入中，字段的值可以是队列管理器在处理 WebSphere MQ API 调用后设置的值，也可以通过出口函数链中的先前出口函数调用设置为值。
 - 在之后 WebSphere MQ API 调用出口函数的输出中，字段的值可以保持不变，也可以由出口函数设置为其他值。
- 出口函数必须使用 ExitResponse 和 ExitResponse2 字段与队列管理器进行通信。
- CompCode 和 Reason 代码字段可通信回应用程序。队列管理器和出口函数可以设置 CompCode 和原因码字段。
- MQXEP 调用将新的原因码返回到调用 MQXEP 的出口函数。但是，出口函数可以将这些新的原因码转换为现有应用程序和新应用程序可以理解的任何现有原因码。
- 除了 CompCode 和 Reason 之外，每个出口函数原型都具有与 API 函数相似的参数，具有额外的 indirection 级别。
- API 出口可以发出 MQI 调用 (MQDISC 除外)，但这些 MQI 调用本身不会调用 API 出口。

请注意，无论应用程序是在服务器上还是在客户机上，您都无法预测 API 出口调用的顺序。API 出口 BEFORE 调用可能不会紧跟 AFTER 调用。

BEFORE 调用可以后跟另一个 BEFORE 调用。例如：

```

MQCTL 之前
BEFORE 回调
在 MQPUT 之前
AFTER MQPUT
AFTER 回调
AFTER MQCTL

```

或

```

在 XAOpen 之前
在 MQCONNX 之前
AFTER MQCONNX
在 XAOpen 之后

```

在客户机上，有一个出口可以修改 MQCONN 或 MQCONNX 调用的行为，称为 PreConnect 出口。PreConnect 出口可以修改 MQCONN 或 MQCONNX 调用上的任何参数，包括队列管理器名称。客户机首先调用此出口，然后调用 MQCONN 或 MQCONNX 调用。请注意，只有初始 MQCONN 或 MQCONNX 调用会调用 API 出口；任何后续重新连接调用都无效。

执行环境

通常，来自出口函数的所有错误都将使用 MQAXP 中的 ExitResponse 和 ExitResponse2 字段传达回出口处理程序。

这些错误将依次转换为 MQCC_* 和 MQRC_* 值，并在 CompCode 和 Reason 字段中传达回应用程序。但是，在出口处理程序逻辑中迂到的任何错误都将作为 CompCode 和 Reason 字段中的 MQCC_* 和 MQRC_* 值传达回应用程序。

如果 MQ_TERM_EXIT 函数返回错误：

- MQDISC 调用已发生
- 没有其他机会来驱动之后 MQ_TERM_EXIT 出口函数 (从而执行出口执行环境清除)
- 未执行出口执行环境清除

无法卸载出口，因为它可能仍在使用中。此外，对于前出口成功的出口链中进一步向下的其他已注册出口，将按相反顺序进行驱动。

设置出口执行环境

在处理显式 MQCONN 或 MQCONNX 调用时，出口处理逻辑会在调用出口初始化函数 (MQ_INIT_EXIT) 之前设置出口执行环境。出口执行环境设置涉及装入出口，获取存储器以及初始化出口参数结构。还会分配出口配置句柄。

如果在此阶段发生错误，那么 MQCONN 或 MQCONNX 调用将失败，并返回 CompCode MQCC_FAILED 和下列其中一个原因码：

MQRC_API_EXIT_LOAD_ERROR

尝试装入 API 出口模块失败。

MQRC_API_EXIT_NOT_FOUND

在 API 出口模块中找不到 API 出口函数。

MQRC_STORAGE_NOT_AVAILABLE

尝试初始化 API 出口函数的执行环境失败，因为没有足够的存储空间可用。

MQRC_API_EXIT_INIT_ERROR

初始化 API 出口函数的执行环境时迂到错误。

清除出口执行环境

在处理显式 MQDISC 调用或作为应用程序结束结果的隐式断开连接请求时，出口处理逻辑可能需要在调用出口终止函数 (MQ_TERM_EXIT) (如果已注册) 之后清除出口执行环境。

清除出口执行环境涉及释放出口参数结构的存储器，可能删除先前装入到内存中的任何模块。

如果在此阶段发生错误，那么显式 MQDISC 调用将失败，并带有 CompCode MQCC_FAILED 和以下原因码 (在隐式断开连接请求中未突出显示错误)：

MQRC_API_EXIT_TERM_ERROR

关闭 API 出口函数的执行环境时迂到错误。在 MQ_TERM* API 出口函数调用之前或之后，出口不应从 MQDISC 返回任何故障。

客户机上的 API 出口

客户机使用 PreConnect 出口来修改 MQCONN 和 MQCONNX 调用的行为，并且不支持 API 出口属性。

PreConnect 出口

在客户机上，可以使用 PreConnect 出口从中央存储库 (例如 LDAP 服务器) 中查找通道定义。

PreConnect 出口还可以修改 MQCONN 或 MQCONNX 调用本身上的任何参数或所有参数，例如队列管理器名称。

对于客户机应用程序，必须在 API 出口之前调用 PreConnect 出口，因为仅当已知队列管理器的名称时才会调用 MQCONN 或 MQCONNX API 出口，并且此名称可由 PreConnect 出口更改。

请注意，只有初始 MQCONN 或 MQCONNX 调用才会调用该出口。

API 出口属性

在服务器上，API 出口可以在初始化时注册 MQXEPO 结构。MQXEPO 结构包含 ExitProperties 字段，该字段详细描述了出口感兴趣的属性组。这将产生单独的消息属性句柄，出口可与任何应用程序消息属性句柄分开处理。

在客户机上，不支持 API 出口属性。如果尝试在客户机上注册属性组名，那么该函数将失败，原因码为 MQRC_EXIT_PROPS_NOT_SUPPORTED。

回退-MQ_BACK_EXIT

MQ_BACK_EXIT 提供回退出函数以执行前和之后回退处理。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_BACK 来注册前和之后回退调用出口函数。

此函数的接口为：

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

CompCode (MQLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```
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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

您的出口必须与以下 C 函数原型匹配：

```

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 */

```

开始-MQ_BEGIN_EXIT

MQ_BEGIN_EXIT 提供了用于执行前和之后 MQBEGIN 调用处理的开始退出函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_BEGIN 来注册前和之后 MQBEGIN 调用出口函数。

此函数的接口为：

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
              &Reason)
```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pBegin 选项 (PMQBO)-输入/输出

用于开始选项的指针。

CompCode (MQLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```

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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```

MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);

```

您的出口必须与以下 C 函数原型匹配：

```

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 */

```

回调-MQ_CALLBACK_EXIT

MQ_CALLBACK_EXIT 提供了用于执行前和之后回调处理的退出函数。使用带有退出原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_CALLBACK 来注册前和之后回调调用退出函数。

此函数的接口为：

```

MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
                  &pBuffer, &pMQCBCContext)

```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构

ExitContext (MQAXC)-输入/输出

出口上下文结构

Hconn (MQHCONN)-输入/输出

连接句柄

pMsgDesc

消息描述符

pGetMsgOpts

用于控制 MQGET 的操作的选项

pBuffer

要包含消息数据的区域

pMQCBCContext

回调的上下文数据

C 语言调用

队列管理器以逻辑方式定义以下变量：

```

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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```

MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
               &pContext);

```

您的出口必须与以下 C 函数原型匹配:

```
void MQENTRY MQ_CALLBACK_EXIT (
    PMQAXP    pExitParms;    /* Exit parameter structure */
    PMQAXC    pExitContext;  /* Exit context structure */
    PMQHCONN  pHconn;        /* Connection handle */
    PPMQMD    ppMsgDesc;     /* Message descriptor */
    PPMQGM0   ppGetMsgOpts;  /* Options that define the operation of the consumer */
    PPMQVOID  ppBuffer;      /* Area to contain the message data */
    PPMQCBC   ppContext;     /* Context data for the callback */
)
```

使用说明

1. 在调用使用者之前以及在使用者的使用者函数完成之后调用回调出口。虽然 MQMD 和 MQGMO 结构是可更改的, 但更改前出口中的值不会重新驱动从队列中检索消息, 因为消息已从队列中移除以传递到使用者函数

管理回调函数-MQ_CB_EXIT

MQ_CB_EXIT 提供了用于执行前和之后 MQCB 调用的出口函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_CB 来注册前和之后 MQCB 调用出口函数。

此函数的接口为:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构

ExitContext (MQAXC)-输入/输出

出口上下文结构

Hconn (MQHCONN)-输入/输出

连接句柄

操作 (MQLONG)-输入/输出

操作值

pCallback 描述 (PMQCBD)-输入/输出

回调描述符

Hobj (MQHOBJ)-输入/输出

对象句柄

pMsg 描述 (PMQMD)-输入/输出

消息描述符

pGetMsgOpts (PMQGM0)-输入/输出

用于控制 MQCB 操作的选项

CompCode (MQLONG)-输入/输出

完成代码

原因 (MQLONG)-输入/输出

原因码限定 CompCode

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
PMQGM0   pGetMsgOpts; /* Options that define the operation of the consumer */
```

```

PMQLONG  CompCode;      /* Completion code.
PMQLONG) Reason;      /* Reason code qualifying CompCode.

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```

MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
            &pGetMsgOpts, &CompCode, &Reason);

```

您的出口必须与以下 C 函数原型匹配：

```

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 */

```

关闭-MQ_CLOSE_EXIT

MQ_CLOSE_EXIT 提供了用于执行前和之后 MQCLOSE 调用处理的关闭出口函数。使用具有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_CLOSE 来注册前和之后 MQCLOSE 调用出口函数。

此函数的接口为：

```

MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
              &Options, &CompCode, &Reason)

```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pHobj (PMQHOBJ)-输入

指向对象句柄的指针。

选项 (MQLONG)-输入/输出

关闭选项。

CompCode (MQLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码为 MQCC_FAILED，那么出口函数可以将原因码字段设置为任何有效的 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：


```

MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;  /* Exit context structure */
MQHCONN    Hconn;       /* Connection handle */
PMQHOBJS  pHobj;       /* Ptr to object handle */
MQLONG    Options;     /* Close options */
MQLONG    CompCode;    /* Completion code */
MQLONG    Reason;      /* Reason code */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```

MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);

```

您的出口必须与以下 C 函数原型匹配：

```

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 */
PPMHOBJS   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 */

```

落实-MQ_CMIT_EXIT

MQ_CMIT_EXIT 提供了用于执行前和之后落实处理的落实退出函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_CMIT 来注册前和之后落实调用出口函数。

如果落实操作失败，并且事务已回退，那么 MQCMIT 调用将失败，并返回 MQCC_WARNING 和 MQRC_BACKED_OUT。这些返回码和原因码将传递到任何之后 MQCMIT 出口函数中，以指示已回退工作单位。

此函数的接口为：

```

MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)

```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

CompCode (MQLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING, 出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

您的出口必须与以下 C 函数原型匹配:

```
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 */
```

使用说明

1. 此处描述的 MQ_GET_EXIT 函数接口用于 MQXF_GET 出口函数和 [第 1000 页](#)的『MQXF_DATA_CONV_ON_GET』出口函数。

为这两个出口函数定义了单独的入口点, 因此要拦截两个 MQXEP 调用, 必须使用两次; 对于此调用, 请使用函数标识 MQXF_GET。

由于 MQXF_GET 和 MQXF_DATA_CONV_ON_GET 的 MQ_GET_EXIT 接口相同, 因此可以将单个出口函数用于这两个函数; MQAXP 结构中的 *Function* 字段指示已调用哪个出口函数。或者, 可以使用 MQXEP 调用来注册这两种情况的不同出口函数。

连接和连接扩展-MQ_CONNX_EXIT

MQ_CONNX_EXIT 提供:

- 用于执行前和之后 MQCONN 处理的连接出口函数
- 用于执行前和之后 MQCONNX 处理的连接扩展出口函数

针对 MQCONN 和 MQCONNX 调用出口函数调用此处描述的同接口。

当消息通道代理程序 (MCA) 响应入站客户机连接时, MCA 可以在客户机状态完全已知之前进行连接并进行大量 WebSphere MQ API 调用。这些 API 调用根据 MCA 程序本身 (例如, 在 MQAXC 的 UserId 和 ConnectionName 字段中) 使用 MQAXC 调用 API 出口函数。

当 MCA 响应后续入站客户机 API 调用时, MQAXC 结构基于入站客户机, 相应地设置 UserId 和 ConnectionName 字段。

应用程序在 MQCONN 或 MQCONNX 调用上设置的队列管理器名称将传递到底层连接调用。在 MQ_CONNX_EXIT 之前更改队列管理器名称的任何尝试都不会产生任何影响。

使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_CONN 和 MQXF_CONNX 来注册前和之后 MQCONN 和 MQCONNX 调用出口函数。

由于 MQXR_BEFORE 不得发出任何 WebSphere MQ API 调用而调用的 MQ_CONNX_EXIT 出口, 因为此时尚未设置正确的环境。

MQ_CONNX_EXIT 无法从要为其调用的连接的 API 出口调用 MQDISC。此限制适用于客户机和服务器 API 出口。

MQCONN 和 MQCONNX 的接口完全相同:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,  
                &pHconn, &CompCode, &Reason);
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

pQMgr 名称 (PMQCHAR)-输入

指向 MQCONNX 调用上提供的队列管理器名称的指针。出口不得在 MQCONN 或 MQCONNX 调用上更改此名称。

pConnectOpts (PMQCNO)-输入/输出

指向用于控制 MQCONNX 调用的操作的选项的指针。

请参阅第 283 页的『MQCNO - 连接选项』, 以了解详细信息。

对于出口函数 MQXF_CONN, pConnect 选项指向缺省连接选项结构 (MQCNO_DEFAULT)。

pHconn (PMQHCONN)-输入

指向连接句柄的指针。

CompCode (MQLONG)-输入/输出

完成代码, 有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

警告 (部分完成)

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK, 那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING, 出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,  
                &pHconn, &CompCode, &Reason);
```

您的出口必须与以下 C 函数原型匹配:

```

void MQENTRY MQ_CONNX_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PPMQCHAR    ppMgrName,      /* Address of ptr to queue manager name */
PPMQCNO     ppConnectOpts,   /* Address of ptr to connection options */
PMQHCONN    ppHconn,        /* Address of ptr to connection handle */
PMLONG      pCompCode,      /* Address of completion code */
PMLONG      pReason);       /* Address of reason code qualifying
                             completion code */

```

使用说明

1. 此处描述的 MQ_CONNX_EXIT 函数接口用于 MQCONN 调用和 MQCONNX 调用。但是，为这两个调用定义了单独的入口点。要拦截这两个调用，必须至少使用两次 MQXEP 调用，一次使用函数标识 MQXF_CONN，再次使用 MQXF_CONNX。

由于 MQ_CONNX_EXIT 接口对于 MQCONN 和 MQCONNX 相同，因此可以将单个出口函数用于这两个调用；MQAXP 结构中的 *Function* 字段指示正在进行的调用。或者，MQXEP 调用可用于为这两个调用注册不同的出口函数。

2. 当消息通道代理程序 (MCA) 响应入站客户机连接时，MCA 可以在客户机状态完全已知之前发出大量 MQ 调用。这些 MQ 调用将导致使用 MQAXC 结构调用 API 出口函数，该结构包含与 MCA 相关的数据，而不包含与客户机相关的数据 (例如，用户标识和连接名称)。但是，一旦客户机状态完全已知，后续 MQ 调用将导致使用 MQAXC 结构中的相应客户机数据调用 API 出口函数。
3. 在队列管理器执行任何参数验证之前，将调用所有 MQXR_BEFORE 出口函数。因此，参数可能无效 (包括参数地址的无效指针)。

在队列管理器执行任何授权检查之前，将调用 MQ_CONNX_EXIT 函数。

4. 出口函数不得更改在 MQCONN 或 MQCONNX 调用上指定的队列管理器的名称。如果出口函数更改了名称，那么将取消定义结果。
5. MQ_CONNX_EXIT 的 MQXR_BEFORE 出口函数不能发出除 MQXEP 以外的 MQ 调用。

控制回调-MQ_CTL_EXIT

MQ_CTL_EXIT 提供预订请求出口函数以执行前和之后控制回调处理。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_CTL 来注册前和之后控制回调调用出口函数。

此函数的接口为：

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

其中，参数为：

Hconn (MQHCONN)-输入/输出

连接句柄。

操作 (PMLONG) 输入/输出

在为指定对象句柄定义的回调上正在处理的操作

ControlOpts (MQCTLO) 输入/输出

用于控制 MQCTL 的操作的选项

CompCode (PMLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (PMLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK, 那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING, 出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

您的出口必须与以下 C 函数原型匹配:

```
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 */
```

断开连接-MQ_DISC_EXIT

MQ_DISC_EXIT 提供了用于执行前和之后 MQDISC 出口处理的断开连接出口函数。使用具有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_DISC 来注册前和之后 MQDISC 调用出口函数。

此函数的接口为

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

pHconn (PMQHCONN)-输入

指向连接句柄的指针。

对于之前的 MQDISC 调用, 此字段的值为下列其中一项:

- 在 MQCONN 或 MQCONNX 调用上返回的连接句柄
- 零, 用于特定于环境的适配器已连接到队列管理器的环境
- 由先前出口函数调用设置的值

对于后 MQDISC 调用, 此字段的值为零或由先前的出口函数调用设置的值。

CompCode (MQLONG)-输入/输出

完成代码, 有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
PMQHCONN   pHconn;        /* Ptr to Connection handle */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;        /* Reason code */
```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

您的出口必须与以下 C 函数原型匹配：

```
void MQENTRY MQ_DISC_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQHCONN    ppHconn,       /* Address of ptr to connection handle */
PMQLONG     pCompCode,     /* Address of completion code */
PMQLONG     pReason);      /* Address of reason code qualifying
                             completion code */
```

获取-MQ_GET_EXIT

MQ_GET_EXIT 提供获取出口函数以执行前和之后 MQGET 调用处理。

有两个函数标识：

1. 将 MQXF_GET 与出口原因 MQXR_BEFORE 和 MQXR_AFTER 配合使用，以注册前和之后 MQGET 调用出口函数。
2. 有关使用 MQXF_DATA_CONV_ON_GET 函数标识的信息，请参阅 [第 1000 页的『MQXF_DATA_CONV_ON_GET』](#)。

此函数的接口为：

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

Hobj (MQHOBJ)-输入/输出

对象句柄。

pMsg 描述 (PMQMD)-输入/输出

指向消息描述符的指针。

pGetMsgOpts (PMQGMO)-输入/输出

用于获取消息选项的指针。

BufferLength (MQLONG)-输入/输出

消息缓冲区长度。

pBuffer (PMQBYTE)-输入/输出

指向消息缓冲区的指针。

pData 长度 (PMQLONG)-输入/输出

指向数据长度字段的指针。

CompCode (MQLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```
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 */
```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

您的出口必须与以下 C 函数原型匹配：

```
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 */

```

使用说明

1. 此处描述的 MQ_GET_EXIT 函数接口用于 MQXF_GET 出口函数和 [第 1000 页的『MQXF_DATA_CONV_ON_GET』](#) 出口函数。

为这两个出口函数定义了单独的入口点，因此要拦截两个 MQXEP 调用，必须使用两次；对于此调用，请使用函数标识 MQXF_GET。

由于 MQXF_GET 和 MQXF_DATA_CONV_ON_GET 的 MQ_GET_EXIT 接口相同，因此可以将单个出口函数用于这两个函数；MQAXP 结构中的 *Function* 字段指示已调用哪个出口函数。或者，可以使用 MQXEP 调用来注册这两种情况的不同出口函数。

MQXF_DATA_CONV_ON_GET

请参阅 [MQ_GET_EXIT](#)，以获取有关此调用的接口以及样本 C 语言声明的信息。

使用说明

如果已注册，那么当消息到达应用程序时，但在发生任何数据转换之前，将调用此入口点。如果 API 出口需要在将消息传递到数据转换之前执行处理（例如解密或解压），那么这可能很有用。如果需要，出口可能会导致通过返回 MQXCC_SUPPRESS_FUNCTION 而绕过数据转换；有关更多信息，请参阅 [MQAXP](#) 结构。

在客户机上注册此入口点会导致在客户机上本地执行数据转换。因此，为了正确操作，可能需要在客户机上安装应用程序转换出口。请注意，MQXF_DATA_CONV_ON_GET 也用于异步使用。

使用 MQ_GET_EXIT 调用时，请使用带有退出原因 MQXR_BEFORE 的 MQXF_DATA_CONV_ON_GET 来注册之前 MQGET 数据转换退出函数。

MQXF_DATA_CONV_ON_GET 没有 MQXR_AFTER 出口函数；MQXF_GET 的 MQXR_AFTER 出口函数提供在数据转换后进行出口处理所需的功能。

为 MQ_GET_EXIT 调用定义了单独的入口点，因此要拦截两个出口函数，必须使用 MQXEP 调用两次；对于此调用，请使用函数标识 MQXF_DATA_CONV_ON_GET。

由于 MQXF_GET 和 MQXF_DATA_CONV_ON_GET 的 MQ_GET_EXIT 接口相同，因此可以将单个出口函数用于这两个函数；MQAXP 结构中的 *Function* 字段指示已调用哪个出口函数。或者，可以使用 MQXEP 调用来注册这两种情况的不同出口函数。

初始化-MQ_INIT_EXIT

MQ_INIT_EXIT 提供连接级别初始化，通过将 MQAXP 中的 ExitReason 设置为 MQXR_CONNECTION 来指示。

在初始化期间，请注意以下事项：

- MQ_INIT_EXIT 函数调用 MQXEP 以注册 WebSphere MQ API 动词及其感兴趣的 ENTRY 和 EXIT 点。
- 出口不需要拦截所有 WebSphere MQ API 动词。仅当已注册兴趣时，才会调用出口函数。
- 可以在初始化出口时获取该出口要使用的存储器。
- 如果对此函数的调用失败，那么调用该函数的 MQCONN 或 MQCONNX 调用也将失败，并带有依赖于 MQAXP 中 ExitResponse 字段值的 CompCode 和 Reason。
- MQ_INIT_EXIT 出口不得发出 WebSphere MQ API 调用，因为此时尚未设置正确的环境。
- 如果 MQ_INIT_EXIT 由于 MQXCC_FAILED 而失败，那么队列管理器将从使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 调用它的 MQCONN 或 MQCONNX 调用中返回。
- 如果队列管理器在调用第一个 MQ_INIT_EXIT 之前初始化 API EXIT 函数执行环境时迁到错误，那么队列管理器将从调用 MQ_INIT_EXIT 和 MQRC_API_EXIT_INIT_ERROR 的 MQCONN 或 MQCONNX 调用返回。

MQ_INIT_EXIT 的接口为:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

CompCode (MQLONG)-输入/输出

指向完成代码的指针, 其有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

指向限定完成代码的原因码的指针。

如果完成代码是 MQCC_OK, 那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING, 出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

返回到应用程序的 CompCode 和 Reason 取决于 MQAXP 中 ExitResponse 字段的值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;        /* Reason code */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

您的出口必须与以下 C 函数原型匹配:

```
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 */
```

使用说明

1. MQ_INIT_EXIT 函数可以发出 MQXEP 调用, 以针对要拦截的特定 MQ 调用注册出口函数的地址。不需要拦截所有 MQ 调用, 也不需要同时拦截 MQXR_BEFORE 和 MQXR_AFTER 调用。例如, 出口套件可以选择仅拦截 MQPUT 的 MQXR_BEFORE 调用。
2. 要由出口套件中的出口函数使用的存储器可由 MQ_INIT_EXIT 函数获取。或者, 当需要时, 出口函数可以在调用时获取存储器。但是, 应该在终止出口套件之前释放所有存储器; MQ_TERM_EXIT 函数可以释放存储器, 或者释放先前调用的出口函数。

3. 如果 MQ_INIT_EXIT 在 MQAXP 的 *ExitResponse* 字段中返回 MQXCC_FAILED，或者以某种其他方式失败，那么导致调用 MQ_INIT_EXIT 的 MQCONN 或 MQCONNEX 调用也将失败，并将 *CompCode* 和 *Reason* 参数设置为相应的值。
4. MQ_INIT_EXIT 函数无法发出除 MQXEP 以外的 MQ 调用。

查询-MQ_INQ_EXIT

MQ_INQ_EXIT 提供了用于执行前和之后 MQINQ 调用处理的查询出口函数。使用具有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_INQ 来注册前和之后 MQINQ 调用出口函数。

此函数的接口为:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

其中，参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

Hobj (MQHOBJ)-输入

对象句柄。

SelectorCount (MQLONG)-输入

选择器计数

pSelectors (PMQLONG)-输入/输出

指向选择器值数组的指针。

IntAttr 计数 (MQLONG)-输入

整数属性的计数。

pIntAttrs (PMQLONG)-输入/输出

指向整数属性值数组的指针。

CharAttr 长度 (MQLONG)-输入/输出

字符属性数组长度。

pCharAttrs (PMQCHAR)-输入/输出

指向字符属性数组的指针。

CompCode (MQLONG)-输入/输出

完成代码，有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

您的出口必须与以下 C 函数原型匹配:

```
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 */
PMQHOBJS  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 */
PPMQLONG  ppCharAttrs,     /* Address of ptr to character attributes array */
PMQLONG   pCompCode,       /* Address of completion code */
PMQLONG   pReason);        /* Address of reason code qualifying completion
                             code */
```

打开-MQ_OPEN_EXIT

MQ_OPEN_EXIT 提供开放式出口函数以执行前和之后 MQOPEN 调用处理。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_OPEN 来注册前和之后 MQOPEN 调用出口函数。

此函数的接口为

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pObj 描述 (PMQOD)-输入/输出

指向对象描述符的指针。

选项 (MQLONG)-输入/输出

打开选项。

pHobj (PMQHOBJS)-输入

指向对象句柄的指针。

CompCode (MQLONG)-输入/输出

完成代码, 有效值为:

MQCC_OK
成功完成。

MQCC_WARNING
部分完成

MQCC_FAILED
通话失败

原因 (MQLONG)-输入/输出
限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE
(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
PMQOD      pObjDesc;      /* Ptr to object descriptor */
MQLONG     Options;       /* Open options */
PMQHOBJS   pHobj;        /* Ptr to object handle */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;       /* Reason code */
```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason);
```

您的出口必须与以下 C 函数原型匹配：

```
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 */
    PPMQHOBJS   ppHobj,         /* Address of ptr to object handle */
    PMQLONG     pCompCode,       /* Address of completion code */
    PMQLONG     pReason);       /* Address of reason code qualifying
                                completion code */
```

放置-MQ_PUT_EXIT

MQ_PUT_EXIT 提供了用于执行前和之后 MQPUT 调用处理的 put 出口函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_PUT 来注册前和之后 MQPUT 调用出口函数。

此函数的接口为：

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

其中，参数为：

ExitParms (MQAXP)-输入/输出
出口参数结构。

ExitContext (MQAXC)-输入/输出
出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

Hobj (MQHOBJ)-输入/输出

对象句柄。

pMsg 描述 (PMQMD)-输入/输出

指向消息描述符的指针。

pPutMsgOpts (PMQPMO)-输入/输出

用于放置消息选项的指针。

BufferLength (MQLONG)-输入/输出

消息缓冲区长度。

pBuffer (PMQBYTE)-输入/输出

指向消息缓冲区的指针。

CompCode (MQLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```

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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```

MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)

```

您的出口必须与以下 C 函数原型匹配：

```

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      pMsgDesc,    /* Address of ptr to message descriptor */
PPMQPMO     ppPutMsgOpts, /* Address of ptr to put message options */
PMQLONG     pBufferLength, /* Address of message buffer length */
PPMQBYTE    pBuffer,     /* Address of ptr to message buffer */
PMQLONG     pCompCode,   /* Address of completion code */

```

```
PMQLONG          pReason);          /* Address of reason code qualifying
                                   completion code */
```

使用说明

- 队列管理器生成的报告消息跳过正常调用处理。因此，MQ_PUT_EXIT 函数或 MQPUT1 函数无法拦截此类消息。但是，正常处理消息通道代理程序生成的报告消息，因此可以通过 MQ_PUT_EXIT 函数或 MQ_PUT1_EXIT 函数进行拦截。为了确保拦截 MCA 生成的所有报告消息，应同时使用 MQ_PUT_EXIT 和 MQ_PUT1_EXIT。

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT 提供了用于执行前和之后 MQPUT1 调用处理的仅放置一条消息出口函数。使用具有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_PUT1 来注册前和之后 MQPUT1 调用出口函数。

此函数的接口为:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

其中，参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pObj 描述 (PMQOD)-输入/输出

指向对象描述符的指针。

pMsg 描述 (PMQMD)-输入/输出

指向消息描述符的指针。

pPutMsgOpts (PMQPMO)-输入/输出

用于放置消息选项的指针。

BufferLength (MQLONG)-输入/输出

消息缓冲区长度。

pBuffer (PMQBYTE)-输入/输出

指向消息缓冲区的指针。

CompCode (MQLONG)-输入/输出

完成代码，有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

您的出口必须与以下 C 函数原型匹配:

```
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 */
PMQBYTE     ppBuffer,     /* Address of ptr to message buffer */
PMQLONG     pCompCode,    /* Address of completion code */
PMQLONG     pReason);    /* Address of reason code qualifying
                           completion code */
```

设置-MQ_SET_EXIT

MQ_SET_EXIT 提供了用于执行前和之后 MQSET 调用处理的集出口函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_SET 来注册前和之后 MQSET 调用出口函数。

此函数的接口为:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

Hobj (MQHOBJ)-输入

对象句柄。

SelectorCount (MQLONG)-输入

选择器计数

pSelectors (PMQLONG)-输入/输出

指向选择器值数组的指针。

IntAttr 计数 (MQLONG)-输入

整数属性的计数。

pIntAttrs (PMQLONG)-输入/输出

指向整数属性值数组的指针。

CharAttr 长度 (MQLONG)-输入/输出

字符属性数组长度。

pCharAttrs (PMQCHAR)-输入/输出

指向字符属性值的指针。

CompCode (MQLONG)-输入/输出

完成代码，有效值为：

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING，出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```
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 */
```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

您的出口必须与以下 C 函数原型匹配：

```
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 */
```

状态-MQ_STAT_EXIT

MQ_STAT_EXIT 提供了用于执行前和之后 MQSTAT 调用处理的状态出口函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_STAT 来注册前和之后 MQSTAT 调用出口函数。

此函数的接口为:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus  
              &CompCode, &Reason)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

类型 (MQLONG)-输入

要检索的状态信息的类型。

pStatus (PMQSTS)-输出

指向状态缓冲区的指针。

CompCode (MQLONG)-输入/输出

完成代码, 有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK, 那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING, 出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

您的出口必须与以下 C 函数原型匹配:

```
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 */
```

终止-MQ_TERM_EXIT

MQ_TERM_EXIT 提供连接级别终止, 使用函数标识 MQXF_TERM 和 ExitReason MQXR_CONNECTION 注册。如果已注册, 那么将对每个断开连接请求调用一次 MQ_TERM_EXIT。

作为终止的一部分, 可以释放出口不再需要的存储器, 并且可以执行所需的任何清除。

如果 MQ_TERM_EXIT 因 MQXCC_FAILED 而失败, 那么队列管理器将从使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 调用它的 MQDISC 返回。

如果队列管理器在调用最后一个 MQ_TERM_EXIT 之后终止 API 出口函数执行环境时迁到错误，那么队列管理器将从使用 MQCC_FAILED 和 MQRC_API_EXIT_TERM_ERROR 调用 MQ_TERM_EXIT 的 MQDISC 调用中返回

此函数的接口为:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

其中，参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

CompCode (MQLONG)-输入/输出

完成代码，有效值为:

MQCC_OK

成功完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK，那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码为 MQCC_FAILED，那么出口函数可以将原因码字段设置为任何有效的 MQRC_* 值。

返回到应用程序的 CompCode 和 Reason 取决于 MQAXP 中 ExitResponse 字段的值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;        /* Reason code */
```

然后，队列管理器以逻辑方式调用出口，如下所示:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

您的出口必须与以下 C 函数原型匹配:

```
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 */
```

使用说明

1. MQ_TERM_EXIT 函数是可选的。如果没有要执行的终止处理，那么出口套件不需要注册终止出口。

如果属于出口套件的函数在连接期间获取资源，那么 MQ_TERM_EXIT 函数是释放这些资源 (例如，释放动态获取的存储器) 的方便点。

2. 如果在发出 MQDISC 调用时注册了 MQ_TERM_EXIT 函数，那么将在调用所有 MQDISC 出口函数之后调用该出口函数。

- 如果 MQ_TERM_EXIT 在 MQAXP 的 *ExitResponse* 字段中返回 MQXCC_FAILED，或者以某种其他方式失败，那么导致调用 MQ_TERM_EXIT 的 MQDISC 调用也将失败，并将 *CompCode* 和 *Reason* 参数设置为相应的值。

注册预订-MQ_SUB_EXIT

MQ_SUB_EXIT 提供了用于执行前和之后预订重新注册处理的出口函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_SUB 来注册前和之后预订 registrationcall 出口函数。

此函数的接口为:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入/输出

连接句柄。

pSub 描述-输入/输出

属性选择器的数组。

pHobj -输入/输出

对象句柄

pHsub (MQHOBJ) 输入/输出

预订句柄

CompCode (MQLONG)-输入/输出

完成代码, 有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK, 那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING, 出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
PMQSD      pSubDesc;      /* Subscription descriptor */
PMQHOBJS   pHobj;        /* Object Handle */
PMQHOBJS   pHsub;        /* Subscription handle */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;        /* Reason code qualifying completion code */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
&CompCode, &Reason);
```

您的出口必须与以下 C 函数原型匹配:

```
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 */
```

预订请求-MQ_SUBRQ_EXIT

MQ_SUBRQ_EXIT 提供预订请求出口函数以执行前和之后预订请求处理。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_SUBRQ 来注册前和之后预订请求调用出口函数。

此函数的接口为:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
&CompCode, &Reason)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入/输出

连接句柄。

pHsub (MQHOBJ) 输入/输出

预订句柄

操作 (MQLONG) 输入/输出

操作

pSubRqOpts (MQSRO) 输入/输出

CompCode (MQLONG)-输入/输出

完成代码, 有效值为:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

通话失败

原因 (MQLONG)-输入/输出

限定完成代码的原因码。

如果完成代码是 MQCC_OK, 那么唯一的有效值是:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果完成代码是 MQCC_FAILED 或 MQCC_WARNING, 出口函数可以将原因码字段设置为任何有效 MQRC_* 值。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```

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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```

MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason);

```

您的出口必须与以下 C 函数原型匹配：

```

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 提供 xa_close 出口函数以在 xa_close 处理之前和之后执行。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XACLOSE 来注册前和后 xa_close 调用出口函数。

此函数的接口为：

```

XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)

```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXa_info (PMQCHAR)-输入/输出

特定于实例的资源管理器信息。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```

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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配：

```
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 提供 xa_commit 出口函数以在 xa_commit 处理之前和之后执行。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XACOMMIT 来注册前和后 xa_commit 调用出口函数。

此函数的接口为：

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```
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 */
```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配：

```
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 提供 xa_complete 出口函数以在 xa_complete 处理之前和之后执行。使用带有退出原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XACOMPLETE 来注册前和后 xa_complete 调用退出函数。

此函数的接口为:

```

XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags,
&XARetCode)

```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pHandle (PMQLONG)-输入/输出

指向异步操作的指针。

pRetVal (PMQLONG)-输入/输出

异步操作的返回值。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```

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 */

```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```

XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags,
&XARetCode);

```

您的出口必须与以下 C 函数原型匹配:

```

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 提供了要在 xa_end 处理之前和之后执行的 xa_end 出口函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XAEND 来注册前和后 xa_end 调用出口函数。

此函数的接口为:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配:

```
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_算了-XA_FORGET_EXIT

XA_FORGET_EXIT 提供要在 xa_算了处理前后执行的 xa_遗忘出口函数。使用带有退出原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XAFORGET 来注册前和后 xa_遗忘调用退出函数。

此函数的接口为:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配:

```
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 提供 xa_open 出口函数以在 xa_open 处理之前和之后执行。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XAOPEN 来注册前和后 xa_open 调用出口函数。

此函数的接口为:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXa_info (PMQCHAR)-输入/输出

特定于实例的资源管理器信息。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配:

```
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 提供了要在 xa_prepare 处理之前和之后执行的 xa_prepare 出口函数。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XAPREPARE 来注册 xa_prepare 前后的调用出口函数。

此函数的接口为:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配:

```
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 提供 `xa_recover` 出口函数以在 `xa_recover` 处理前后执行。使用带有退出原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XARECOVER 来注册 `xa_recover` 调用退出函数前后的函数。

此函数的接口为:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

计数 (MQLONG)-输入/输出

XID 数组中的最大 XID 数

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配：

```

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 */
    MQPTR   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 提供 xa_rollback 出口函数以在 xa_rollback 处理之前和之后执行。使用带有退出原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XAROLLBACK 来注册 xa_rollback 调用前后的退出函数。

此函数的接口为：

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```

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 */

```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配:

```
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 提供 xa_start 出口函数以在 xa_start 处理之前和之后执行。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_XASTART 来注册前和后 xa_start 调用出口函数。

此函数的接口为:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

其中, 参数为:

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```
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 */
```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配:

```
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 提供了一个 ax_reg 出口函数，用于在 ax_reg 处理前后执行。使用带有退出原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_AXREG 来注册 ax_reg 调用前后的退出函数。

此函数的接口为：

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Hconn (MQHCONN)-输入

连接句柄。

pXID (MQPTR)-输入/输出

事务分支标识。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量：

```
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 */
```

然后，队列管理器以逻辑方式调用出口，如下所示：

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配：

```
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 提供了一个 ax_unreg 出口函数，用于在 ax_unreg 处理前后执行。使用带有出口原因 MQXR_BEFORE 和 MQXR_AFTER 的函数标识 MQXF_AXUNREG 来注册 ax_unreg 调用前后的出口函数。

此函数的接口为：

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

其中，参数为：

ExitParms (MQAXP)-输入/输出

出口参数结构。

ExitContext (MQAXC)-输入/输出

出口上下文结构。

Rmid (MQLONG)-输入/输出

资源管理器标识。

标志 (MQLONG)-输入/输出

资源管理器选项。

XARetCode (MQLONG)-输入/输出

来自 XA 调用的响应。

C 语言调用

队列管理器以逻辑方式定义以下变量:

```

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 */

```

然后, 队列管理器以逻辑方式调用出口, 如下所示:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

您的出口必须与以下 C 函数原型匹配:

```

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 */

```

有关调用出口函数的常规信息

本主题提供了一些一般指导, 以帮助您规划出口, 特别是与处理错误和意外事件相关的出口。

退出失败

如果出口函数在破坏性的, 超出同步点的 MQGET 调用之后, 但在将消息传递到应用程序之前异常终止, 那么出口处理程序可以从故障中恢复, 并将控制权传递给应用程序。

在这种情况下, 消息可能会丢失。这类似于应用程序在从队列接收消息后立即失败时发生的情况。

MQGET 调用可能使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 完成。

如果 *before* API 调用出口函数异常终止, 那么出口处理程序可以从故障中恢复并将控制传递到应用程序, 而无需处理 API 调用。在此情况下, 出口函数必须恢复它拥有的任何资源。

如果正在使用链式出口, 那么可以自行驱动已成功驱动的任何之前 API 调用出口的之后 API 调用出口。API 调用可能失败, 带有 MQCC_FAILED 和 MQRC_API_EXIT_ERROR。

出口函数的错误处理示例

下图显示了可能发生错误的点 (eN)。它只是一个示例, 说明出口的行为方式, 应该与下表一起读取。在此示例中, 在每次 API 调用之前和之后都将调用两个出口函数, 以显示具有链式出口的行为。

| 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  <--
        -->
e8      before MQ_OPEN_EXIT 1
e9      before MQ_OPEN_EXIT 2
e10     --> MQOPEN
e11     after  MQ_OPEN_EXIT 2
e12     after  MQ_OPEN_EXIT 1
e12
MQPUT   <--
        -->
e13     before MQ_PUT_EXIT 1
e14     before MQ_PUT_EXIT 2
e15     --> MQPUT
e16     after  MQ_PUT_EXIT 2
e17     after  MQ_PUT_EXIT 1
e17
MQCLOSE <--
        -->
e18     before MQ_CLOSE_EXIT 1
e19     before MQ_CLOSE_EXIT 2
e20     --> MQCLOSE
e21     after  MQ_CLOSE_EXIT 2
e22     after  MQ_CLOSE_EXIT 1
e22
MQDISC  <--
        -->
e23     before MQ_DISC_EXIT 1
e24     before MQ_DISC_EXIT 2
e25     --> MQDISC
e26     after  MQ_DISC_EXIT 2
e27     after  MQ_DISC_EXIT 1
e27
        <--
end

```

下表列出了要在每个错误点执行的操作。仅覆盖了部分错误点，因为此处显示的规则可以适用于所有其他错误点。它是在每种情况下指定预期行为的操作。

表 595: API 出口错误和要执行的相应操作

| Err Pt | 描述 | 操作 |
|--------|---|--|
| e1 | 设置环境设置时出错。 | <ol style="list-style-type: none"> 1. 根据需要撤销环境设置 2. 磁带机无出口功能 3. MQCONN 失败, 带有 MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR |
| e2 | MQ_INIT_EXIT 函数完成时间: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED: <ol style="list-style-type: none"> 1. 清除环境 2. MQCONN 失败, 带有 MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • 对于 MQXCC_* <ol style="list-style-type: none"> 1. 充当 MQXCC_* 和 MQXR2_*¹ 的值 2. 清除环境 |
| e3 | 在 MQ_CONNX_EXIT 1 函数完成之前: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED: <ol style="list-style-type: none"> 1. 磁带机 MQ_TERM_EXIT 函数 2. 清除环境 3. 使用 MQCC_FAILED, MQRC_API_EXIT_ERROR 的 MQCONN 调用失败 • 对于 MQXCC_* <ol style="list-style-type: none"> 1. 充当 MQXCC_* 和 MQXR2_*¹ 的值 2. 驱动器 MQ_TERM_EXIT 函数 (如果需要) 3. 清除环境 (如果需要) |
| e4 | 在 MQ_CONNX_EXIT 2 函数完成之前: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED: <ol style="list-style-type: none"> 1. 磁带机 之后 MQ_CONNX_EXIT 1 函数 2. 磁带机 MQ_TERM_EXIT 函数 3. 清除环境 4. 使用 MQCC_FAILED, MQRC_API_EXIT_ERROR 的 MQCONN 调用失败 • 对于 MQXCC_* <ol style="list-style-type: none"> 1. 充当 MQXCC_* 和 MQXR2_*¹ 的值 2. 驱动器 之后 MQ_CONNX_EXIT 1 函数 (如果未禁止退出) 3. 驱动器 MQ_TERM_EXIT 函数 (如果需要) 4. 清除环境 (如果需要) |
| e5 | MQCONN 调用失败。 | <ol style="list-style-type: none"> 1. 传递 MQCONN CompCode 和原因 2. 驱动器 之后 MQ_CONNX_EXIT 2 函数 (如果前 MQ_CONNX_EXIT 2 成功并且未禁止出口) 3. 驱动器 之后 MQ_CONNX_EXIT 1 函数 (如果前 MQ_CONNX_EXIT 1 成功并且未禁止出口) 4. 磁带机 MQ_TERM_EXIT 函数 5. 清除环境 |

表 595: API 出口错误和要执行的相应操作 (继续)

| Err Pt | 描述 | 操作 |
|--------|--|--|
| e6 | 在 MQ_CONNX_EXIT 2 函数完成后: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED: <ol style="list-style-type: none"> 1. 磁带机 之后 MQ_CONNX_EXIT 1 函数 2. 使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 完成 MQCONN 调用 • 对于 MQXCC_* <ol style="list-style-type: none"> 1. 充当 MQXCC_* 和 MQXR2_ *1 的值 2. 磁带机 之后 MQ_CONNX_EXIT 1 功能 (如果需要) |
| e7 | 在 MQ_CONNX_EXIT 1 函数完成后: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED, 使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 完成 MQCONN 调用 • 对于 MQXCC_*, 充当 MQXCC_* 和 MQXR2_ *1 的值 |
| e8 | 在 MQ_OPEN_EXIT 1 函数完成之前: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED, 使用 MQCC_FAILED 完成 MQOPEN 调用, MQRC_API_EXIT_ERROR • 对于 MQXCC_*, 充当 MQXCC_* 和 MQXR2_ *1 的值 |
| e9 | 在 MQ_OPEN_EXIT 2 函数完成之前: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED: <ol style="list-style-type: none"> 1. 磁带机 之后 MQ_OPEN_EXIT 1 函数 2. 使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 完成 MQOPEN 调用 • 对于 MQXCC_*, 充当 MQXCC_* 和 MQXR2_ *1 的值 |
| e10 | MQOPEN 调用失败 | <ol style="list-style-type: none"> 1. 传递 MQOPEN CompCode 和原因 2. 驱动器 之后 MQ_OPEN_EXIT 2 函数 (如果未禁止退出) 3. 驱动器 之后 MQ_OPEN_EXIT 1 函数 (如果未禁止退出并且未禁止链式出口) |
| e11 | 在 MQ_OPEN_EXIT 2 函数完成后: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED: <ol style="list-style-type: none"> 1. 磁带机 之后 MQ_OPEN_EXIT 1 函数 2. 使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 完成 MQOPEN 调用 • 对于 MQXCC_* <ol style="list-style-type: none"> 1. 充当 MQXCC_* 和 MQXR2_ *1 的值 2. 驱动器 之后 MQ_OPEN_EXIT 1 函数 (如果未禁止退出) |

表 595: API 出口错误和要执行的相应操作 (继续)

| Err Pt | 描述 | 操作 |
|--------|---|---|
| e25 | 在 MQ_DISC_EXIT 2 函数完成后: <ul style="list-style-type: none"> • MQXCC_FAILED • MQXCC_* | <ul style="list-style-type: none"> • 对于 MQXCC_FAILED: <ol style="list-style-type: none"> 1. 磁带机 之后 MQ_DISC_EXIT 1 函数 2. 磁带机 MQ_TERM_EXIT 函数 3. 清除出口执行环境 4. 使用 MQCC_FAILED 和 MQRC_API_EXIT_ERROR 完成 MQDISC 调用 • 对于 MQXCC_* <ol style="list-style-type: none"> 1. 充当 MQXCC_* 和 MQXR2_*¹ 的值 2. 磁带机 MQ_TERM_EXIT 函数 3. 清除出口执行环境 |

注:

1. MQXCC_* 和 MQXR2_* 的值及其相应操作在 [队列管理器如何处理出口函数](#) 中定义。

ExitResponse 字段设置不正确

本主题提供有关当 ExitResponse 字段设置为受支持的值以外的任何值时将发生的情况的信息。

如果 ExitResponse 字段设置为除其中一个受支持的值以外的值，那么以下操作适用:

- 对于之前的 MQCONN 或 MQDISC API 出口函数:
 - 将忽略 ExitResponse2 值。
 - 在调用出口链 (如果有) 中的出口函数 之前 不再执行任何操作; 不会发出 API 调用本身。
 - 对于成功调用的任何 之前 出口，将按反向顺序调用 之后 出口。
 - 如果已注册，那么将驱动链中成功调用的那些 之前 MQCONN 或 MQDISC 出口函数的终止出口函数以在这些出口函数之后进行清除。
 - MQCONN 或 MQDISC 调用失败，发生 MQRC_API_EXIT_ERROR。
- 对于之前的 WebSphere MQ API 出口函数 (MQCONN 或 MQDISC 除外):
 - 将忽略 ExitResponse2 值。
 - 在调用出口链 (如果有) 中的数据转换函数 之前 或 之后 没有进一步的数据转换函数。
 - 对于成功调用的任何 之前 出口，将按反向顺序调用 之后 出口。
 - 不会发出 WebSphere MQ API 调用本身。
 - WebSphere MQ API 调用失败，发生 MQRC_API_EXIT_ERROR。
- 对于后 MQCONN 或 MQDISC API 出口函数:
 - 将忽略 ExitResponse2 值。
 - 在 API 调用之前成功调用的其余出口函数将按反向顺序进行调用。
 - 如果已注册，那么将驱动链中成功调用的 之前 或 之后 MQCONN 或 MQDISC 出口函数的终止出口函数以在出口之后进行清除。
 - 出口返回的 MQCC_WARNING 较严重的 CompCode 和 CompCode 将返回到应用程序。
 - 将 MQRC_API_EXIT_ERROR 的原因返回给应用程序。
 - 已成功发出 WebSphere MQ API 调用。
- 对于之后的 WebSphere MQ API 调用出口函数 (MQCONN 或 MQDISC 除外):
 - 将忽略 ExitResponse2 值。

- 在 API 调用之前成功调用的其余出口函数将按反向顺序进行调用。
- 出口返回的 MQCC_WARNING 较严重的 CompCode 和 CompCode 将返回到应用程序。
- 将 MQRC_API_EXIT_ERROR 的原因返回给应用程序。
- 已成功发出 WebSphere MQ API 调用。
- 对于获取出口函数上的 之前 数据转换:
 - 将忽略 ExitResponse2 值。
 - 在 API 调用之前成功调用的其余出口函数将按反向顺序进行调用。
 - 不转换消息, 未转换的消息将返回到应用程序。
 - 出口返回的 MQCC_WARNING 较严重的 CompCode 和 CompCode 将返回到应用程序。
 - 将 MQRC_API_EXIT_ERROR 的原因返回给应用程序。
 - 已成功发出 WebSphere MQ API 调用。

注: 由于出口存在错误, 因此返回 MQRC_API_EXIT_ERROR 比返回 MQRC_NOT_汇率更好。

如果出口函数将 ExitResponse2 字段设置为除其中一个受支持值以外的值, 那么将改为假定值为 MQXR2_DEFAULT_CONTINUATION。

可安装服务接口参考信息

此主题集合提供可安装服务的参考信息。

函数和数据类型在每个服务类型的组中按字母顺序列出。

函数的显示方式

如何记录可安装服务的功能。

对于每个函数, 都有一个描述, 包括函数标识 (对于 MQZEP)。

参数 按它们必须出现的顺序列出。他们必须都在场。

每个参数名称后跟其数据类型。这些是 [第 214 页的『基本数据类型』](#) 中描述的基本数据类型。

在参数描述之后, 还会给出 C 语言调用。

MQZ_AUTHENTICATE_USER-认证用户

此函数由 MQZAS_VERSION_5 授权服务组件提供, 并由队列管理器调用以认证用户或设置身份上下文。当建立 WebSphere MQ 的用户应用程序上下文时, 将调用此命令。

在连接调用期间, 将在初始化应用程序的用户上下文的点建立应用程序上下文, 并在更改应用程序的用户上下文的每个点建立应用程序上下文。每次发出连接调用时, 都会在 *IdentityContext* 字段中重新获取应用程序的用户上下文信息。

此函数 (针对 MQZEP) 的函数标识为 MQZID_AUTHENTICATE_USER。

语法

MQZ_AUTHENTICATE_USER (*QMgrName*, *SecurityParms*, *ApplicationContext*, *IdentityContext*, *CorrelationPtr*, *ComponentData*, *Continuation*, *CompCode*, *原因*)

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用 该名称。

SecurityParms

类型:MQCSP-输入

安全性参数。与用户标识、密码和认证类型相关的数据。如果将 MQCSP 结构的 AuthenticationType 属性指定为 MQCSP_AUTH_USER_ID_AND_PWD, 那么会将用户标识和密码与 IdentityContext (MQZIC) 参数中的等效字段进行比较, 以确定它们是否匹配。有关更多信息, 请参阅 [第 296 页的『MQCSP-安全性参数』](#)。

在 MQCONN MQI 调用期间, 此参数包含空值或缺省值。

ApplicationContext

类型:MQZAC-输入

应用程序上下文。与调用应用程序相关的数据。请参阅 [MQZAC-应用程序上下文](#) 以获取详细信息。

在每次 MQCONN 或 MQCONNX MQI 调用期间, 都会重新获取 MQZAC 结构中的用户上下文信息。

IdentityContext

类型:MQZIC-输入/输出

身份上下文。在认证用户函数的输入上, 此标识当前身份上下文。认证用户功能可以更改此, 此时队列管理器将采用新的身份上下文。有关 MQZIC 结构的更多详细信息, 请参阅 [MQZIC-身份上下文](#)。

CorrelationPtr

类型:MQPTR-输出

关联指针。指定任何关联数据的地址。此指针随后传递到其他 OAM 调用。

ComponentData

类型:MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保留; 将保留此组件提供的任何函数对其进行的任何更改, 并在下次调用此组件的 s 函数之一时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 ComponentData 长度参数中传递。

延续

类型: MQLONG - 输出

延续标志。可指定以下值:

MQZCI_DEFAULT

依赖于其他组件的延续。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 CompCode。

如果 CompCode 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 CompCode 为 MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, 'X'8F1') 访问服务时发生意外错误。

有关这些原因码的更多信息，请参阅 [原因码](#)。

C 调用

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, &CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

声明 传递到服务的参数，如下所示：

```
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-检查权限

此函数由 MQZAS_VERSION_1 授权服务组件提供，并由队列管理器启动，以检查实体是否有权对指定对象执行特定操作。

此函数 (针对 MQZEP) 的函数标识为 MQZID_CHECK_AUTHORITY。

语法

```
MQZ_CHECK_AUTHORITY( QMgrName, EntityName, EntityType, ObjectName, ObjectType,  
Authority, ComponentData, Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityName

类型: MQCHAR12 - 输入

实体名称。要检查其对对象的权限的实体的名称。字符串的最大长度为 12 个字符; 如果它比它短，那么将用空格填充到右边。名称未以空字符终止。

此实体对于底层安全服务而言并不是必需的。如果未知，那么将使用特殊 **没人** 组 (假定所有实体都属于该组) 的权限进行检查。全空白名称有效，可以通过此方式使用。

EntityType

类型: MQLONG - 输入

实体类型。由 EntityName 指定的实体类型。它必须是下列其中一个值：

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ObjectName

类型: MQCHAR48 - 输入

对象名称。需要访问的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型: MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。 它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

authority

类型: MQLONG-输入

要检查的权限。 如果正在检查一个授权, 那么此字段等于相应的授权操作 (MQZAO_* 常量)。 如果正在检查多个授权, 那么它是相应 MQZAO_* 常量的按位 OR。

以下授权适用于 MQI 调用的使用:

MQZAO_CONNECT

能够使用 MQCONN 调用。

MQZAO_BROWSE

能够将 MQGET 调用与浏览选项配合使用。

这允许在 MQGET 调用上指定 MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR 或 MQGMO_BROWSE_NEXT 选项。

MQZAO_INPUT

校长 能够将 MQGET 调用与输入选项配合使用。

这允许在 MQOPEN 调用上指定 MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE 或 MQOO_INPUT_AS_Q_DEF 选项。

MQZAO_OUTPUT

能够使用 MQPUT 调用。

这允许在 MQOPEN 调用上指定 MQOO_OUTPUT 选项。

MQZAO_INQUIRE

能够使用 MQINQ 调用。

这允许在 MQOPEN 调用上指定 MQOO_INQUIRE 选项。

MQZAO_SET

能够使用 MQSET 调用。

这允许在 MQOPEN 调用上指定 MQOO_SET 选项。

MQZAO_PASS_IDENTITY_CONTEXT

能够传递身份上下文。

这允许在 MQOPEN 调用上指定 MQOO_PASS_IDENTITY_CONTEXT 选项，并在 MQPUT 和 MQPUT1 调用上指定 MQPMO_PASS_IDENTITY_CONTEXT 选项。

MQZAO_PASS_ALL_CONTEXT

能够传递所有上下文。

这允许在 MQOPEN 调用上指定 MQOO_PASS_ALL_CONTEXT 选项，在 MQPUT 和 MQPUT1 调用上指定 MQPMO_PASS_ALL_CONTEXT 选项。

MQZAO_SET_IDENTITY_CONTEXT

能够设置身份上下文。

这允许在 MQOPEN 调用上指定 MQOO_SET_IDENTITY_CONTEXT 选项，并在 MQPUT 和 MQPUT1 调用上指定 MQPMO_SET_IDENTITY_CONTEXT 选项。

MQZAO_SET_ALL_CONTEXT

能够设置所有上下文。

这允许在 MQOPEN 调用上指定 MQOO_SET_ALL_CONTEXT 选项，并在 MQPUT 和 MQPUT1 调用上指定 MQPMO_SET_ALL_CONTEXT 选项。

MQZAO_ALTERNATE_USER_AUTHORITY

能够使用备用用户权限。

这允许在 MQOPEN 调用上指定 MQOO_ALTERNATE_USER_AUTHORITY 选项，并在 MQPUT1 调用上指定 MQPMO_ALTERNATE_USER_AUTHORITY 选项。

MQZAO_ALL_MQI

所有 MQI 授权。

这将启用所有授权。

以下权限适用于队列管理器的管理：

MQZAO_CREATE

能够创建指定类型的对象。

MQZAO_DELETE

能够删除指定的对象。

MQZAO_DISPLAY

能够显示指定对象的属性。

MQZAO_CHANGE

能够更改指定对象的属性。

MQZAO_CLEAR

能够从指定队列中删除所有消息。

MQZAO_AUTHORIZE

能够对指定对象的其他用户进行授权。

MQZAO_CONTROL

能够启动或停止侦听器，服务或非客户机通道对象，以及能够对非客户机通道对象执行 ping 操作。

MQZAO_CONTROL_EXTENDED

能够重置序号，或解决非客户机通道对象上的不确定消息。

MQZAO_ALL_ADMIN

能够设置身份上下文。

除 MQZAO_CREATE 以外的所有管理权限。

以下权限适用于 MQI 的使用和队列管理器的管理:

MQZAO_ALL

除 MQZAO_CREATE 以外的所有权限。

MQZAO_NONE

无授权。

ComponentData

类型:MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存;此组件提供的任何函数对其进行的任何更改都将保留,并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY, 这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

如果对组件的调用失败(即, *CompCode* 返回 MQCC_FAILED), 并且 *Continuation* 参数为 MQZCI_DEFAULT 或 MQZCI_CONTINUE, 那么队列管理器将继续调用其他组件(如果有)。

如果调用成功(即, *CompCode* 返回 MQCC_OK), 那么无论 *Continuation* 的设置如何, 都不会调用任何其他组件。

如果调用失败, 并且 *Continuation* 参数为 MQZCI_STOP, 那么不会调用其他组件, 并且会将错误返回到队列管理器。组件不知道先前的调用, 因此在调用之前, *Continuation* 参数始终设置为 MQZCI_DEFAULT。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息，请参阅 [API 原因码](#)。

C 调用

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                    ObjectType, Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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 - 检查权限 (已扩展)

此函数由 MQZAS_VERSION_2 授权服务组件提供，并且由队列管理器启动，以检查实体是否有权对指定对象执行特定操作。

此函数 (针对 MQZEP) 的函数标识为 MQZID_CHECK_AUTHORITY。

MQZ_CHECK_AUTHORITY_2 类似于 MQZ_CHECK_AUTHORITY，但使用 *EntityData* 参数替换 *EntityName* 参数。

语法

```
MQZ_CHECK_AUTHORITY_2( QMgrName, EntityData, EntityType, ObjectName,  
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason )
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityData

类型: MQZED-输入

实体数据。与对要检查的对象具有权限的实体相关的数据。请参阅第 1082 页的『MQZED-实体描述符』，以了解详细信息。

此实体对于底层安全服务而言并不是必需的。如果未知，那么将使用特殊 **没人** 组 (假定所有实体都属于该组) 的权限进行检查。全空白名称有效，可以通过此方式使用。

EntityType

类型: MQLONG-输入

实体类型。EntityData 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ObjectName

类型:MQCHAR48 - 输入

对象名称。需要访问的对象的名称。字符串的最大长度为 48 个字符;如果它比它短,那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型:MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

authority

类型:MQLONG-输入

要检查的权限。如果正在检查一个授权,那么此字段等于相应的授权操作 (MQZAO_* 常量)。如果正在检查多个授权,那么它是相应 MQZAO_* 常量的按位 OR。

以下授权适用于 MQI 调用的使用:

MQZAO_CONNECT

能够使用 MQCONN 调用。

MQZAO_BROWSE

能够将 MQGET 调用与浏览选项配合使用。

这允许在 MQGET 调用上指定 MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR 或 MQGMO_BROWSE_NEXT 选项。

MQZAO_INPUT

校长 能够将 MQGET 调用与输入选项配合使用。

这允许在 MQOPEN 调用上指定 MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE 或 MQOO_INPUT_AS_Q_DEF 选项。

MQZAO_OUTPUT

能够使用 MQPUT 调用。

这允许在 MQOPEN 调用上指定 MQOO_OUTPUT 选项。

MQZAO_INQUIRE

能够使用 MQINQ 调用。

这允许在 MQOPEN 调用上指定 MQOO_INQUIRE 选项。

MQZAO_SET

能够使用 MQSET 调用。

这允许在 MQOPEN 调用上指定 MQOO_SET 选项。

MQZAO_PASS_IDENTITY_CONTEXT

能够传递身份上下文。

这允许在 MQOPEN 调用上指定 MQOO_PASS_IDENTITY_CONTEXT 选项，并在 MQPUT 和 MQPUT1 调用上指定 MQPMO_PASS_IDENTITY_CONTEXT 选项。

MQZAO_PASS_ALL_CONTEXT

能够传递所有上下文。

这允许在 MQOPEN 调用上指定 MQOO_PASS_ALL_CONTEXT 选项，在 MQPUT 和 MQPUT1 调用上指定 MQPMO_PASS_ALL_CONTEXT 选项。

MQZAO_SET_IDENTITY_CONTEXT

能够设置身份上下文。

这允许在 MQOPEN 调用上指定 MQOO_SET_IDENTITY_CONTEXT 选项，并在 MQPUT 和 MQPUT1 调用上指定 MQPMO_SET_IDENTITY_CONTEXT 选项。

MQZAO_SET_ALL_CONTEXT

能够设置所有上下文。

这允许在 MQOPEN 调用上指定 MQOO_SET_ALL_CONTEXT 选项，并在 MQPUT 和 MQPUT1 调用上指定 MQPMO_SET_ALL_CONTEXT 选项。

MQZAO_ALTERNATE_USER_AUTHORITY

能够使用备用用户权限。

这允许在 MQOPEN 调用上指定 MQOO_ALTERNATE_USER_AUTHORITY 选项，并在 MQPUT1 调用上指定 MQPMO_ALTERNATE_USER_AUTHORITY 选项。

MQZAO_ALL_MQI

所有 MQI 授权。

这将启用所有授权。

以下权限适用于队列管理器的管理：

MQZAO_CREATE

能够创建指定类型的对象。

MQZAO_DELETE

能够删除指定的对象。

MQZAO_DISPLAY

能够显示指定对象的属性。

MQZAO_CHANGE

能够更改指定对象的属性。

MQZAO_CLEAR

能够从指定队列中删除所有消息。

MQZAO_AUTHORIZE

能够对指定对象的其他用户进行授权。

MQZAO_CONTROL

能够启动或停止侦听器，服务或非客户机通道对象，以及能够对非客户机通道对象执行 ping 操作。

MQZAO_CONTROL_EXTENDED

能够重置序号，或解决非客户机通道对象上的不确定消息。

MQZAO_ALL_ADMIN

能够设置身份上下文。

除 MQZAO_CREATE 以外的所有管理权限。

以下权限适用于 MQI 的使用和队列管理器的管理：

MQZAO_ALL

除 MQZAO_CREATE 以外的所有权限。

MQZAO_NONE

无授权。

ComponentData

类型：MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存；此组件提供的任何函数对其进行的任何更改都将保留，并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型：MQLONG - 输出

按组件设置的连续指示符。可以指定以下值：

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY，这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型：MQLONG - 输出

完成代码。它必须是下列其中一个值：

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK：

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED：

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息，请参阅 [API 原因码](#)。

C 调用

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,  
                        ObjectName, ObjectType, Authority, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-检查用户是否具有特权

此函数由 MQZAS_VERSION_6 授权服务组件提供，并且由队列管理器调用以确定指定用户是否为特权用户。

此函数 (针对 MQZEP) 的函数标识为 MQZID_CHECK_PRIVILEGED。

语法

```
MQZ_CHECK_PRIVILEGED( QMgrName, EntityData, EntityType, ComponentData,  
Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityData

类型: MQZED-输入

实体数据。与要检查的实体相关的数据。有关更多信息，请参阅第 1082 页的『MQZED-实体描述符』。

EntityType

类型: MQLONG-输入

实体类型。EntityData 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数对其进行的任何更改都将保留，并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY, 这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

如果对组件的调用失败 (即, *CompCode* 返回 MQCC_FAILED), 并且 *Continuation* 参数为 MQZCI_DEFAULT 或 MQZCI_CONTINUE, 那么队列管理器将继续调用其他组件 (如果有)。

如果调用成功 (即, *CompCode* 返回 MQCC_OK), 那么无论 *Continuation* 的设置如何, 都不会调用任何其他组件。

如果调用失败, 并且 *Continuation* 参数为 MQZCI_STOP, 那么不会调用其他组件, 并且会将错误返回到队列管理器。组件不知道先前的调用, 因此在调用之前, *Continuation* 参数始终设置为 MQZCI_DEFAULT。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_PRIVILEGED

(2584, X'A18') 此用户不是特权用户标识。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') 服务的实体未知。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
```

```
ComponentData, &Continuation,  
&CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-复制所有权限

此功能由授权服务组件提供。它由队列管理器启动，以将当前对一个引用对象生效的所有权限复制到另一个对象。

此函数(针对 MQZEP)的函数标识为 MQZID_COPY_ALL_AUTHORITY。

语法

```
MQZ_COPY_ALL_AUTHORITY(QMgrName, RefObjectName, ObjectName, ObjectType,  
ComponentData, Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

RefObject 名称

类型: MQCHAR48 - 输入

引用对象名。要复制其权限的引用对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

ObjectName

类型: MQCHAR48 - 输入

对象名称。要为其设置访问权的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

ObjectType

类型: MQLONG-输入

对象类型。 *RefObjectName* 和 *ObjectName* 指定的实体类型。它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

ComponentData

类型:MQBYTE ×ComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存;此组件提供的任何函数对其进行的任何更改都将保留,并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 ComponentData 长度参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY, 这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_REF_OBJECT

(2294, X'8F6') 引用对象未知。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,  
                          ComponentData, &Continuation, &CompCode,  
                          &Reason);
```

传递到服务的参数声明如下:

```
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-删除权限

此功能由授权服务组件提供，并由队列管理器启动以删除与指定对象关联的所有权限。

此函数 (针对 MQZEP) 的函数标识为 MQZID_DELETE_AUTHORITY。

语法

```
MQZ_DELETE_AUTHORITY( QMgrName, ObjectName, ObjectType, ComponentData,  
Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

ObjectName

类型: MQCHAR48 - 输入

对象名称。要删除其访问权的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型: MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

ComponentData

类型:MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存;此组件提供的任何函数对其进行的任何更改都将保留,并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 ComponentData 长度参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY, 这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,  
                      &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-枚举权限数据

此函数由 MQZAS_VERSION_4 授权服务组件提供, 并且由队列管理器重复启动, 以检索与第一次调用时指定的选择标准相匹配的所有权限数据。

此函数 (针对 MQZEP) 的函数标识为 MQZID_ENUMERATE_AUTHORITY_DATA。

语法

```
MQZ_ENUMERATE_AUTHORITY_DATA( QMgrName, StartEnumeration, Filter,  
AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength, ComponentData,  
Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

StartEnumeration

类型: MQLONG-输入

指示调用是否可以启动枚举的标志。这指示调用是可以启动权限数据的枚举, 还是继续先前对 MQZ_ENUMERATE_AUTHORITY_DATA 的调用所启动的权限数据的枚举。该值是下列其中一个值:

MQZSE_START

开始枚举。使用此值启动调用以启动权限数据的枚举。 *Filter* 参数指定要用于选择由此调用和后续调用返回的权限数据的选择标准。

MQZSE_CONTINUE

继续枚举。使用此值启动调用以继续枚举权限数据。在此情况下, 将忽略 *Filter* 参数, 并且可以将其指定为空指针 (选择标准由将 *StartEnumeration* 设置为 MQZSE_START 的调用所指定的 *Filter* 参数确定)。

Filter

类型: MQZAD-输入

过滤器。如果 *StartEnumeration* 是 MQZSE_START, 那么 *Filter* 指定要用于选择要返回的权限数据的选择条件。如果 *Filter* 是空指针, 那么不使用选择标准, 即返回所有权限数据。请参阅 [第 1080 页的『MQZAD-权限数据』](#) 以获取可使用的选择条件的详细信息。

如果 *StartEnumeration* 为 MQZSE_CONTINUE, 那么将忽略 *Filter*, 并且可以将其指定为空指针。

AuthorityBuffer 长度

类型: MQLONG-输入

AuthorityBuffer 的长度。这是 *AuthorityBuffer* 参数的长度 (以字节计)。权限缓冲区必须足够大, 以容纳要返回的数据。

AuthorityBuffer

类型: MQZAD-输出

权限数据。这是返回权限数据的缓冲区。缓冲区必须足够大, 以容纳 MQZAD 结构, MQZED 结构以及定义的最长实体名称和最长域名。

注: 注: 此参数定义为 MQZAD, 因为 MQZAD 始终在缓冲区启动时出现。但是, 如果将缓冲区声明为 MQZAD, 那么缓冲区将太小-它必须比 MQZAD 大, 以便它可以容纳 MQZAD, MQZED 以及实体和域名。

AuthorityData 长度

类型: MQLONG - 输出

AuthorityBuffer 中返回的数据的长度。如果权限缓冲区太小, 那么 *AuthorityDataLength* 将设置为所需的缓冲区长度, 并且调用将返回完成代码 MQCC_FAILED 和原因码 MQRC_BUFFER_LENGTH_ERROR。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 ComponentData 长度参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_ENUMERATE_AUTHORITY_DATA, 这与 MQZCI_CONTINUE 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') 缓冲区长度参数无效。

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') 无可用数据。

MQRC_SERVICE_ERROR
(2289, X'8F1') 访问服务时发生意外错误。

有关这些原因码的更多信息，请参阅 [API 原因码](#)。

C 调用

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,  
                               AuthorityBufferLength,  
                               &AuthorityBuffer,  
                               &AuthorityDataLength, ComponentData,  
                               &Continuation, &CompCode,  
                               &Reason);
```

传递到服务的参数声明如下:

```
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-可用用户

此函数由 MQZAS_VERSION_5 授权服务组件提供，并由队列管理器启动以释放关联的已分配资源。

当应用程序在所有用户上下文 (例如 MQDISC MQI 调用期间) 下完成运行时，将启动此命令。

此函数 (针对 MQZEP) 的函数标识为 MQZID_FREE_USER。

语法

```
MQZ_FREE_USER( QMgrName, FreeParms, ComponentData, Continuation, CompCode,  
Reason )
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

FreeParms

类型: MQZFP-输入

可用参数。包含与要释放的资源相关的数据的结构。有关详细信息，请参阅第 1085 页的『MQZFP-可用参数』。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 ComponentData 长度参数中传递。

延续

类型: MQLONG - 输出

延续标志。可以指定以下值:

MQZCI_DEFAULT

依赖于其他组件的延续。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-获取权限

此函数由 MQZAS_VERSION_1 授权服务组件提供, 并且由队列管理器启动, 以检索实体具有的访问指定对象的权限, 包括 (如果实体是主体) 主体所属的组所拥有的权限。来自通用概要文件的权限包含在返回的权限集中。

此函数 (针对 MQZEP) 的函数标识为 MQZID_GET_AUTHORITY。

语法

`MQZ_GET_AUTHORITY(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)`

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityName

类型: MQCHAR12 - 输入

实体名称。要检索其对象访问权的实体的名称。字符串的最大长度为 12 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

EntityType

类型: MQLONG-输入

实体类型。 *EntityName* 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ObjectName

类型: MQCHAR48 - 输入

对象名称。要检索其访问权的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型: MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

authority

类型: MQLONG-输入

实体的权限。如果实体具有一个权限,那么此字段等于相应的授权操作 (MQZAO_* 常量)。如果它具有多个权限,那么此字段是相应 MQZAO_* 常量的按位 OR。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存;此组件提供的任何函数对其进行的任何更改都将保留,并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_GET_AUTHORITY, 这与 MQZCI_CONTINUE 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') 服务的实体未知。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                  ObjectType, &Authority, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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 - 获取权限 (已扩展)

此函数由 MQZAS_VERSION_2 授权服务组件提供, 并且由队列管理器启动以检索实体具有的访问指定对象的权限。

此函数 (针对 MQZEP) 的函数标识为 MQZID_GET_AUTHORITY。

MQZ_GET_AUTHORITY_2 类似于 MQZ_GET_AUTHORITY, 但将 *EntityName* 参数替换为 *EntityData* 参数。

语法

```
MQZ_GET_AUTHORITY_2( QMgrName, EntityData, EntityType, ObjectName, ObjectType,  
Authority, ComponentData, Continuation, CompCode, Reason )
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityData

类型: MQZED-输入

实体数据。与要检索其对象授权的实体相关的数据。请参阅第 1082 页的『MQZED-实体描述符』, 以了解详细信息。

EntityType

类型: MQLONG-输入

实体类型。EntityData 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ObjectName

类型: MQCHAR48 - 输入

对象名称。要检索其实体权限的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR , 那么此名称与 *QMgrName* 相同。

ObjectType

类型 :MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。 它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

authority

类型 :MQLONG-输入

实体的权限。 如果实体具有一个权限, 那么此字段等于相应的授权操作 (MQZAO_ * 常量)。 如果它具有多个权限, 那么此字段是相应 MQZAO_ * 常量的按位 OR。

ComponentData

类型 :MQBYTE xComponentData 长度-输入/输出

组件数据。 此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。 可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY , 这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。 它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') 服务的实体未知。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

语法

MQZ_GET_AUTHORITY_2(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

C 调用

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                    ObjectType, &Authority, ComponentData,
                    &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-获取显式权限

此函数由 MQZAS_VERSION_1 授权服务组件提供, 并由队列管理器启动, 以检索指定组具有的访问指定对象的权限 (但没有 **nobody** 组的其他权限) 或指定主体的主组具有的访问指定对象的权限。

在 UNIX 平台上, 对于内置 WebSphere MQ 对象权限管理器 (OAM), 返回的权限仅由主体的主组拥有。

此函数 (针对 MQZEP) 的函数标识为 MQZID_GET_EXPLICIT_AUTHORITY。

语法

`MQZ_GET_EXPLICIT_AUTHORITY(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)`

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityName

类型: MQCHAR12 - 输入

实体名称。要检索其对象访问权的实体的名称。字符串的最大长度为 12 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

EntityType

类型: MQLONG-输入

实体类型。 *EntityName* 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL
校长

MQZAET_GROUP
组。

ObjectName

类型: MQCHAR48 - 输入

对象名称。要检索其实体权限的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型: MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。它必须是下列其中一个值:

MQOT_AUTH_INFO
认证信息。

MQOT_CHANNEL
通道。

MQOT_CLNTCONN_CHANNEL
客户机连接通道。

MQOT_LISTENER
侦听器。

MQOT_NAMELIST
NAMELIST.

MQOT_PROCESS
process definition.

MQOT_Q
队列。

MQOT_Q_MGR
队列管理器。

MQOT_服务
服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

authority

类型: MQLONG-输入

实体的权限。如果实体具有一个权限,那么此字段等于相应的授权操作 (MQZAO_* 常量)。如果它具有多个权限,那么此字段是相应 MQZAO_* 常量的按位 OR。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存;此组件提供的任何函数对其进行的任何更改都将保留,并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_GET_AUTHORITY, 这与 MQZCI_CONTINUE 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') 服务的实体未知。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,  
                             ObjectName, ObjectType, &Authority,  
                             ComponentData, &Continuation,  
                             &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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 - 获取显式权限 (已扩展)

此函数由 MQZAS_VERSION_2 授权服务组件提供, 并由队列管理器启动, 以检索指定组必须访问指定对象的权限 (但没有 没人 组的其他权限) 或指定主体的主组必须访问指定对象的权限。

此函数 (针对 MQZEP) 的函数标识为 MQZID_GET_EXPLICIT_AUTHORITY。

MQZ_GET_EXPLICIT_AUTHORITY_2 类似于 MQZ_GET_EXPLICIT_AUTHORITY, 但将 *EntityName* 参数替换为 *EntityData* 参数。

语法

```
MQZ_GET_EXPLICIT_AUTHORITY_2( QMgrName, EntityData, EntityType, ObjectName,  
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityData

类型: MQZED-输入

实体数据。与要检索其对该对象的授权的实体相关的数据。请参阅第 1082 页的『MQZED-实体描述符』, 以了解详细信息。

EntityType

类型: MQLONG-输入

实体类型。 *EntityData* 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ObjectName

类型: MQCHAR48 - 输入

对象名称。要检索其实体权限的对象名称。字符串的最大长度为 48 个字符;如果它比它短,那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型: MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

authority

类型: MQLONG-输入

实体的权限。如果实体具有一个权限,那么此字段等于相应的授权操作 (MQZAO_* 常量)。如果它具有多个权限,那么此字段是相应 MQZAO_* 常量的按位 OR。

ComponentData

类型: MQBYTE ×ComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存;此组件提供的任何函数对其进行的任何更改都将保留,并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY, 这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') 服务的实体未知。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,  
                               ObjectName, ObjectType, &Authority,  
                               ComponentData, &Continuation,  
                               &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-初始化授权服务

此功能由授权服务组件提供, 并由队列管理器在组件配置期间启动。期望调用 MQZEP 以向队列管理器提供信息。

此函数 (针对 MQZEP) 的函数标识为 MQZID_INIT_AUTHORITY。

语法

```
MQZ_INIT_AUTHORITY(Hconfig, Options, QMgrName, ComponentDataLength,  
ComponentData, Version, CompCode, Reason)
```

参数

Hconfig

类型 :MQHCONFIG-输入

配置句柄。此句柄表示正在初始化的特定组件。它将由组件在使用 MQZEP 函数调用队列管理器时使用。

选项

类型 :MQLONG-输入

初始化选项。它必须是下列其中一个值:

MQZIO_PRIMARY

主初始化。

MQZIO_SECONDARY

辅助初始化。

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

ComponentData 长度

类型 :MQLONG-输入

组件数据的长度。 *ComponentData* 区域的长度 (以字节计)。此长度在组件配置数据中定义。

ComponentData

类型 :MQBYTE xComponentData 长度-输入/输出

组件数据。这将在调用组件主初始化函数之前初始化为全零。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数 (包括初始化函数) 对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

版本

类型 :MQLONG-输入/输出

版本号。在输入到初始化函数时, 这将标识队列管理器支持的最高版本号。如果需要, 初始化函数必须将其更改为它支持的接口版本。如果返回时队列管理器不支持组件返回的版本, 那么它将调用组件 MQZ_TERM_AUTHORITY 函数, 并且不再使用此组件。

支持以下值:

MQZAS_VERSION_1

版本 1。

MQZAS_VERSION_2

版本 2。

MQZAS_VERSION_3

版本 3。

MQZAS_VERSION_4

版本 4。

MQZAS_VERSION_5

版本 5。

MQZAS_VERSION_6

版本 6。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值：

MQCC_OK
成功完成。

MQCC_FAILED
调用失败。

原因

类型：MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE
(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') 初始化失败，原因未定义。

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息，请参阅 [API 原因码](#)。

C 调用

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,  
                    ComponentData, &Version, &CompCode,  
                    &Reason);
```

传递到服务的参数声明如下：

```
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-INQUIRE 授权服务

此函数由 MQZAS_VERSION_5 授权服务组件提供，并由队列管理器启动以查询受支持的功能。

在使用多个服务组件的情况下，将按服务组件的安装顺序逆向调用服务组件。

此函数（针对 MQZEP）的函数标识为 MQZID_INQUIRE。

语法

```
MQZ_INQUIRE( QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs,  
CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation,  
CompCode, Reason )
```

参数

QMgrName

类型：MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度；该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

SelectorCount

类型 :MQLONG-输入

选择器数。 *Selectors* 参数中提供的选择器数。

该值必须在 0 到 256 范围内。

选择器

类型: MQLONGxSelector 计数-输入

选择器的数组。 每个选择器都标识一个必需属性, 并且必须是下列其中一项:

- MQIACF_INTERFACE_VERSION (整数)
- MQIACF_USER_ID_SUPPORT (整数)
- MQCACF_SERVICE_COMPONENT (字符)

可以按任何顺序指定选择器。 数组中的选择器数由 *SelectorCount* 参数指示。

选择器标识的整数属性在 *IntAttrs* 参数中的返回顺序与它们在 *Selectors* 中的显示顺序相同。

选择器标识的字符属性在 *CharAttrs* 参数中返回的顺序与它们在 *Selectors* 中的显示顺序相同。

IntAttrCount

类型 :MQLONG-输入

IntAttrs 参数中提供的整数属性数。

该值必须在 0 到 256 范围内。

IntAttrs

类型 :MQLONG xIntAttr 计数-输出

整数属性。 整数属性的数组。 返回整数属性的顺序与 *Selectors* 数组中相应整数选择器的顺序相同。

CharAttr 计数

类型 :MQLONG-输入

字符属性缓冲区的长度。 *CharAttrs* 参数的长度 (以字节计)。

该值必须至少是所请求字符属性的长度总和。 如果未请求任何字符属性, 那么零是有效值。

CharAttrs

类型 :MQLONG xCharAttr 计数-输出

字符属性缓冲区。 包含字符属性的缓冲区, 并置在一起。 字符属性的返回顺序与 *Selectors* 数组中相应的字符选择器的返回顺序相同。

缓冲区的长度由 *CharAttrCount* 参数给出。

SelectorReturned

类型 :MQLONG xSelectorCount -输入

返回了选择器。 用于标识从选择器参数中的选择器所请求的集合返回的属性的值数组。 此数组中的值数由 *SelectorCount* 参数指示。 数组中的每个值都与选择器数组中相应位置的选择器相关。 每个值都是下列其中一项:

已返回 MQZSL_退回

已返回 *Selectors* 参数中相应选择器所请求的属性。

MQZSL_NOT_返还

未返回 *Selectors* 参数中相应选择器所请求的属性。

使用所有值作为 *MQZSL_NOT_RETURNED* 初始化数组。 当授权服务组件返回属性时, 它会将数组中的相应值设置为 *MQZSL_NOT_RETURNED* 。 这允许对其进行查询调用的任何其他授权服务组件标识已返回的属性。

ComponentData

类型 :MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY, 这与 MQZCI_STOP 具有相同的效果。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_WARNING

部分完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

没有足够的空间用于字符属性。

MQRC_INT_COUNT_TOO_SMALL

没有足够的空间用于整数属性。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR

选择器的数目无效。

MQRC_SELECTOR_ERROR

属性选择器无效。

已超过 MQRC_SELECTOR_LIMIT_AUTHORIZED

指定的选择器过多。

MQRC_INT_ATTR_COUNT_ERROR

整数属性的数目无效。

MQRC_INT_ATTRS_ARRAY_ERROR

整数属性数组无效。

MQRC_CHAR_ATTR_LENGTH_ERROR

字符属性数无效。

MQRC_CHAR_ATTRS_ERROR

字符属性字符串无效。

MQRC_SERVICE_ERROR

(2289, 'X'8F1') 访问服务时发生意外错误。

有关这些原因码的更多信息，请参阅 [API 原因码](#)。

C 调用

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
              &IntAttrs, CharAttrLength, &CharAttrs,
              SelectorReturned, ComponentData, &Continuation,
              &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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];       /* Character 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-刷新所有授权

此函数由 MQZAS_VERSION_3 授权服务组件提供，并且由队列管理器调用以刷新组件内部持有的权限列表。

此函数 (针对 MQZEP) 的函数标识为 MQZID_REFRESH_CACHE (8L)。

语法

```
MQZ_REFRESH_CACHE( QMgrName, ComponentData, Continuation, CompCode, Reason )
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保留; 此组件提供的任何函数对其进行的任何更改都将保留，并在下次调用此组件的某个函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY，其效果与 MQZCI_STOP 相同。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型：MQLONG - 输出

完成代码。它必须是下列其中一个值：

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型：MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

C 调用

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

按如下所示声明参数：

```
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-设置权限

此函数由 MQZAS_VERSION_1 授权服务组件提供，并且由队列管理器启动以设置实体访问指定对象的权限。

此函数 (针对 MQZEP) 的函数标识为 MQZID_SET_AUTHORITY。

注: 此函数将覆盖任何现有权限。要保留任何现有权限，必须使用此功能再次设置这些权限。

语法

```
MQZ_SET_AUTHORITY( QMgrName, EntityName, EntityType, ObjectName, ObjectType,  
Authority, ComponentData, Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityName

类型: MQCHAR12 - 输入

实体名称。要检索其对象访问权的实体的名称。字符串的最大长度为 12 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

EntityType

类型: MQLONG - 输入

实体类型。 *EntityName* 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ObjectName

类型: MQCHAR48 - 输入

对象名称。需要访问的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型: MQLONG - 输入

对象类型。 *ObjectName* 指定的实体类型。它必须是下列其中一个值:

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

authority

类型: MQLONG - 输入

实体的权限。如果设置了一个权限，那么此字段等于相应的授权操作 (MQZAO_* 常量)。如果设置了多个权限，那么此字段是相应 MQZAO_* 常量的按位 OR。

ComponentData

类型:MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存;此组件提供的任何函数对其进行的任何更改都将保留，并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。可以指定以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_GET_AUTHORITY，这与 MQZCI_CONTINUE 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') 服务的实体未知。

有关这些原因码的更多信息，请参阅 [API 原因码](#)。

C 调用

```
MQZ_SET_AUTHORITY (QMgzName, EntityName, EntityType, ObjectName,  
                  ObjectType, Authority, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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 -设置权限 (扩展)

此函数由 MQZAS_VERSION_2 授权服务组件提供,并由队列管理器启动以设置实体访问指定对象所需的权限。

此函数 (针对 MQZEP) 的函数标识为 MQZID_SET_AUTHORITY。

注: 此函数将覆盖任何现有权限。要保留任何现有权限,必须使用此功能再次设置这些权限。

MQZ_SET_AUTHORITY_2 类似于 MQZ_SET_AUTHORITY, 但将 *EntityName* 参数替换为 *EntityData* 参数。

语法

`MQZ_SET_AUTHORITY_2(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)`

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

EntityData

类型: MQZED-输入

实体数据。与要设置对该对象的授权的实体相关的数据。请参阅第 1082 页的『MQZED-实体描述符』, 以了解详细信息。

EntityType

类型: MQLONG-输入

实体类型。 *EntityData* 指定的实体类型。它必须是下列其中一个值:

MQZAET_PRINCIPAL

校长

MQZAET_GROUP

组。

ObjectName

类型: MQCHAR48 - 输入

对象名称。要对其设置实体权限的对象的名称。字符串的最大长度为 48 个字符; 如果它比它短, 那么将用空格填充到右边。名称未以空字符终止。

如果 *ObjectType* 是 MQOT_Q_MGR, 那么此名称与 *QMgrName* 相同。

ObjectType

类型: MQLONG-输入

对象类型。 *ObjectName* 指定的实体类型。 它必须是下列其中一个值：

MQOT_AUTH_INFO

认证信息。

MQOT_CHANNEL

通道。

MQOT_CLNTCONN_CHANNEL

客户机连接通道。

MQOT_LISTENER

侦听器。

MQOT_NAMELIST

NAMELIST.

MQOT_PROCESS

process definition.

MQOT_Q

队列。

MQOT_Q_MGR

队列管理器。

MQOT_服务

服务。

MQOT_TOPIC

主题中查看此版本新增功能的摘要。

authority

类型 :MQLONG-输入

实体的权限。 如果设置了一个权限，那么此字段等于相应的授权操作 (MQZAO_* 常量)。 如果设置了多个权限，那么此字段是相应 MQZAO_* 常量的按位 OR。

ComponentData

类型 :MQBYTE xComponentData 长度-输入/输出

组件数据。 此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数对其进行的任何更改都将保留，并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。 可以指定以下值：

MQZCI_DEFAULT

依赖于队列管理器的延续。

对于 MQZ_CHECK_AUTHORITY，这与 MQZCI_STOP 具有相同的效果。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。 它必须是下列其中一个值：

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') 未获得访问授权。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') 服务的实体未知。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,  
                    ObjectType, Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-终止授权服务

此功能由授权服务组件提供, 并且由队列管理器在不再需要此组件的服务时启动。此函数必须执行组件所需的任何清除。

此函数 (针对 MQZEP) 的函数标识为 MQZID_TERM_AUTHORITY。

语法

```
MQZ_TERM_AUTHORITY( Hconfig, Options, QMgrName, ComponentData, CompCode,  
Reason )
```

参数

Hconfig

类型: MQHCONFIG-输入

配置句柄。此句柄表示正在终止的特定组件。它将由组件在使用 MQZEP 函数调用队列管理器时使用。

选项

类型: MQLONG-输入

终止选项。它必须是下列其中一个值:

MQZTO_PRIMARY

主终止。

MQZTO_SECONDARY

辅助终止。

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用上的 ComponentDataLength 参数中传递。

MQZ_TERM_AUTHORITY 调用完成后, 队列管理器将废弃此数据。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_TERMINATION_FAILED

(2287, X'8FF') 由于未定义的原因而终止失败。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,  
                    &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-删除名称

此功能由名称服务组件提供，并由队列管理器启动以删除指定队列的条目。

此函数（针对 MQZEP）的函数标识为 MQZID_DELETE_NAME。

语法

```
MQZ_DELETE_NAME( QMgrName, QName, ComponentData, Continuation, CompCode,
Reason )
```

参数

QMgrName

类型：MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度；该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息；授权服务接口不要求组件以任何定义的方式使用该名称。

QName

类型：MQCHAR48 - 输入

队列名称。要删除其条目的队列的名称。此名称将用空格填充到参数的完整长度；该名称未以空字符终止。

ComponentData

类型：MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存；此组件提供的任何函数对其进行的任何更改都将保留，并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_NAME 调用上的 ComponentDataLength 参数中传递。

延续

类型：MQLONG - 输出

按组件设置的连续指示符。它必须是下列其中一个值：

MQZCI_DEFAULT

依赖于队列管理器的延续。

MQZCI_STOP

请勿继续使用下一个组件。

对于 **MQZ_DELETE_NAME** 命令，无论 **Continuation** 参数中返回的内容如何，队列管理器都不会尝试启动其他组件。

CompCode

类型：MQLONG - 输出

完成代码。它必须是下列其中一个值：

MQCC_OK

成功完成。

MQCC_WARNING

警告（部分完成）。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_WARNING:

MQRC_UNKNOWN_NAME

(2288, X'8F0') 找不到队列名称。

注: 如果底层服务成功响应此案例, 那么可能无法返回此代码。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,  
&CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-初始化名称服务

此功能由名称服务组件提供, 并由队列管理器在组件配置期间启动。期望调用 MQZEP 以向队列管理器提供信息。

此函数(针对 MQZEP)的函数标识为 MQZID_INIT_NAME。

语法

```
MQZ_INIT_NAME( Hconfig, Options, QMgrName, ComponentDataLength, ComponentData,  
Version, CompCode, Reason)
```

参数

Hconfig

类型: MQHCONFIG-输入

配置句柄。此句柄表示正在初始化的特定组件。它将由组件在使用 MQZEP 函数调用队列管理器时使用。

选项

类型: MQLONG-输入

初始化选项。它必须是下列其中一个值:

MQZIO_PRIMARY

主初始化。

MQZIO_SECONDARY

辅助初始化。

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

ComponentData 长度

类型: MQLONG-输入

组件数据的长度。 *ComponentData* 区域的长度 (以字节计)。此长度在组件配置数据中定义。

ComponentData

类型: MQBYTE ×ComponentData 长度-输入/输出

组件数据。这将在调用组件主初始化函数之前初始化为全零。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数 (包括初始化函数) 对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_AUTHORITY 调用的 *ComponentDataLength* 参数中传递。

版本

类型: MQLONG-输入/输出

版本号。在输入到初始化函数时, 这将标识队列管理器支持的最高版本号。如果需要, 初始化函数必须将其更改为它支持的接口版本。如果返回时队列管理器不支持组件返回的版本, 那么它将调用组件 MQZ_TERM_NAME 函数, 并且不再使用此组件。

支持以下值:

MQZAS_VERSION_1

版本 1。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') 初始化失败, 原因未定义。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,  
               ComponentData, &Version, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-插入名称

此函数由名称服务组件提供，并由队列管理器启动以插入指定队列的条目，其中包含拥有该队列的队列管理器的名称。如果已在服务中定义队列，那么调用将失败。

此函数 (针对 MQZEP) 的函数标识为 MQZID_INSERT_NAME。

语法

```
MQZ_INSERT_NAME( QMgrName, QName, ResolvedQMgrName, ComponentData,  
                Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

QName

类型: MQCHAR48 - 输入

队列名称。要为其插入条目的队列的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

ResolvedQMgrName

类型: MQCHAR48 - 输入

已解析的队列管理器名称。队列解析到的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

ComponentData

类型: MQBYTE ×ComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数 (包括初始化函数) 对其进行的任何更改都将保留，并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_NAME 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG-输入/输出

按组件设置的连续指示符。对于 MQZ_INSERT_NAME，队列管理器不会尝试启动其他组件，无论 *Continuation* 参数中返回的内容是什么。

支持以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_Q_ALREADY_EXISTS

(2290, X'8F2') 队列对象已存在。

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,  
&Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-查找名称

此函数由名称服务组件提供, 并且由队列管理器启动以检索指定队列的拥有队列管理器的名称。

此函数 (针对 MQZEP) 的函数标识为 MQZID_LOOKUP_NAME。

语法

```
MQZ_LOOKUP_NAME( QMgrName, QName, ResolvedQMgrName, ComponentData,  
Continuation, CompCode, Reason)
```

参数

QMgrName

类型: MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

QName

类型: MQCHAR48 - 输入

队列名称。要为其解析条目的队列的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

ResolvedQMgrName

类型: MQCHAR48 - 输出

已解析的队列管理器名称。如果该函数成功完成, 那么这是拥有该队列的队列管理器的名称。

服务组件返回的名称必须在右边用空格填充到参数的完整长度; 该名称不得以空字符终止, 也不得包含前导或嵌入的空格。

ComponentData

类型: MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数 (包括初始化函数) 对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

此数据区的长度由队列管理器在 MQZ_INIT_NAME 调用的 *ComponentDataLength* 参数中传递。

延续

类型: MQLONG - 输出

按组件设置的连续指示符。对于 MQZ_LOOKUP_NAME, 队列管理器指定是否启动另一个名称服务组件, 如下所示:

- 如果 *CompCode* 为 MQCC_OK, 那么不会启动其他组件, 无论 延续中返回的值如何。
- 如果 *CompCode* 不是 MQCC_OK, 那么将启动另一个组件, 除非 *Continuation* 是 MQZCI_STOP。

支持以下值:

MQZCI_DEFAULT

依赖于队列管理器的延续。

MQZCI_CONTINUE

继续下一个组件。

MQZCI_STOP

请勿继续使用下一个组件。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') 访问服务时发生意外错误。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

MQRC_UNKNOWN_Q_NAME

(2288, X'8F0') 找不到队列名称。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,  
&Continuation, &CompCode, &Reason);
```

传递到服务的参数声明如下:

```
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-终止名称服务

此函数由名称服务组件提供, 并且由队列管理器在不再需要此组件的服务时启动。此函数必须执行组件所需的任何清除。

此函数 (针对 MQZEP) 的函数标识为 MQZID_TERM_NAME。

语法

```
MQZ_TERM_NAME( Hconfig, Options, QMgrName, ComponentData, CompCode, Reason )
```

参数

Hconfig

类型:MQHCONFIG-输入

配置句柄。此句柄表示正在终止的特定组件。在使用 MQZEP 函数调用队列管理器时, 组件将使用此参数。

选项

类型:MQLONG-输入

终止选项。它必须是下列其中一个值:

MQZTO_PRIMARY

主终止。

MQZTO_SECONDARY

辅助终止。

QMgrName

类型:MQCHAR48 - 输入

队列管理器名称。调用组件的队列管理器的名称。此名称将用空格填充到参数的完整长度; 该名称未以空字符终止。

队列管理器名称将传递给组件以获取信息; 授权服务接口不要求组件以任何定义的方式使用该名称。

ComponentData

类型:MQBYTE xComponentData 长度-输入/输出

组件数据。此数据由队列管理器代表此特定组件保存; 此组件提供的任何函数 (包括初始化函数) 对其进行的任何更改都将保留, 并在下次调用其中一个组件函数时显示。

组件数据位于可供所有进程访问的共享内存中。

此数据区的长度由队列管理器在 MQZ_INIT_NAME 调用的 *ComponentDataLength* 参数中传递。

MQZ_TERM_NAME 调用完成后, 队列管理器将废弃此数据。

CompCode

类型: MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型: MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_TERMINATION_FAILED

(2287, X'8FF') 由于未定义的原因而终止失败。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 底层服务不可用。

有关这些原因码的更多信息, 请参阅 [API 原因码](#)。

C 调用

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,
                &Reason);
```

传递到服务的参数声明如下:

```
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-应用程序上下文

MQZAC 结构用于 *ApplicationContext* 参数的 MQZ_AUTHENTICATE_USER 调用。此参数指定与调用应用程序相关的数据。

表 1 汇总了结构中的字段。

表 596: MQZAC 中的字段

| 字段 | 描述 |
|---------------------------|--------|
| <u>StrucId</u> | 结构标识 |
| 版本 | 结构版本号 |
| <u>ProcessId</u> | 进程标识 |
| <u>ThreadId</u> | 线程标识 |
| <u>ApplName</u> | 应用程序名称 |
| <u>UserID</u> | 用户标识 |
| <u>EffectiveUser 标识</u> | 有效用户标识 |
| 环境 | 环境 |
| <u>CallerType</u> | 调用者类型 |
| <u>AuthenticationType</u> | 认证类型 |
| <u>BindType</u> | 绑定类型 |

字段

StrucId

类型: MQCHAR4 -输入

结构标识。值如下所示:

MQZAC_STRUC_ID

应用程序上下文结构的标识。

对于 C 编程语言, 还定义了常量 MQZAC_STRUC_ID_ARRAY; 这与 MQZAC_STRUC_ID 具有相同的值, 但是字符数组而不是字符串。

版本

类型: MQLONG-输入

结构版本号。值如下所示:

MQZAC_VERSION_1

Version-1 应用程序上下文结构。常量 MQZAC_CURRENT_VERSION 指定当前版本的版本号。

ProcessId

类型: MQPID-输入

应用程序的进程标识。

ThreadId

类型: MQTID-输入

应用程序的线程标识。

ApplName

类型: MQCHAR28 -输入

应用程序名称。

UserID

类型: MQCHAR12 -输入

用户标识。在 UNIX 系统上, 此字段指定应用程序的实际用户标识。在 Windows 上, 此字段指定应用程序的用户标识。

EffectiveUser 标识

类型: MQCHAR12 -输入

有效用户标识。在 UNIX 系统上，此字段指定应用程序的有效用户标识。在 Windows 上，此字段为空白。

环境

类型: MQLONG-输入

环境。此字段指定从中进行调用的环境。该字段是下列其中一个值:

MQXE_COMMAND_SERVER

命令服务器

MQXE_MQSC

runmqsc 命令解释器

MQXE_MCA

消息通道代理程序 MQXE_OTHER

MQXE_OTHER

未定义的环境

CallerType

类型: MQLONG-输入

调用者类型。此字段指定进行调用的程序的类型。该字段是下列其中一个值:

MQXACT_EXTERNAL

调用是队列管理器的外部调用。

MQXACT_INTERNAL

调用是队列管理器的内部调用。

AuthenticationType

类型: MQLONG-输入

认证类型。此字段指定要执行的认证类型。该字段是下列其中一个值:

MQZAT_INITIAL_CONTEXT

认证调用是由于正在初始化用户上下文。此值在 MQCONN 或 MQCONNX 调用期间使用。

MQZAT_CHANGE_CONTEXT

认证调用是由于正在更改用户上下文。当 MCA 更改用户上下文时，将使用此值。父主题: MQZAC-

BindType

类型: MQLONG-输入

绑定类型。此字段指定正在使用的绑定类型。该字段是下列其中一个值:

MQCNO_FASTPATH_BINDING

快速路径绑定。

MQCNO_SHARED_BINDING

共享绑定。

MQCNO_ISOLATED_BINDING

隔离绑定。

C 声明

按如下所示声明结构的字段:

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
    MQCHAR4    StructId;           /* 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-权限数据

MQZAD 结构在 MQZ_ENUMERATE_AUTHORITY_DATA 调用上用于两个参数，一个输入和一个输出。

- MQZAD 用于输入到调用的 *Filter* 参数。此参数指定要用于选择调用返回的权限数据的选择标准。
- MQZAD 还用于调用输出的 *AuthorityBuffer* 参数。此参数指定概要文件名称，对象类型和实体的一个组合的权限。

表 1。汇总了结构中的字段。

| 字段 | 描述 |
|-------------------------------|--------|
| StrucId | 结构标识 |
| 版本 | 结构版本号 |
| ProfileName | 进程标识 |
| ObjectType | 线程标识 |
| 权限 | 应用程序名称 |
| EntityDataPtr | 用户标识 |
| EntityType | 环境 |
| 选项 | 调用者类型 |

字段

StrucId

类型: MQCHAR4 -输入

结构标识。值如下所示:

MQZAC_STRUC_ID

应用程序上下文结构的标识。

对于 C 编程语言，还定义了常量 MQZAC_STRUC_ID_ARRAY; 这与 MQZAC_STRUC_ID 具有相同的值，但是字符数组而不是字符串。

版本

类型 :MQLONG-输入

结构版本号。值如下所示:

MQZAC_VERSION_1

Version-1 应用程序上下文结构。常量 MQZAC_CURRENT_VERSION 指定当前版本的版本号。

以下常量指定当前版本的版本号:

MQZAD_CURRENT_VERSION

权限数据结构的当前版本。

ProfileName

类型: MQCHAR48 - 输入

概要文件名称。

对于 过滤器 参数，此字段是需要权限数据的概要文件名称。如果该名称完全为空白，直到字段末尾或第一个空字符，那么将返回所有概要文件名称的权限数据。

对于 *AuthorityBuffer* 参数，此字段是与指定选择标准匹配的概要文件的名称。

ObjectType

类型 :MQLONG-输入

对象类型。

对于 过滤器 参数, 此字段是需要权限数据的对象类型。如果值为 MQOT_ALL, 那么将返回所有对象类型的权限数据。

对于 *AuthorityBuffer* 参数, 此字段是 *ProfileName* 参数标识的概要文件所应用于的对象类型。

值为下列其中一项; 对于 *Filter* 参数, 值 MQOT_ALL 也有效:

MQOT_AUTH_INFO

认证信息

MQOT_CHANNEL

通道

MQOT_CLNTCONN_CHANNEL

客户机连接通道

MQOT_LISTENER

侦听器

MQOT_NAMELIST

名称列表

MQOT_PROCESS

进程定义

MQOT_Q

队列

MQOT_Q_MGR

队列管理器

MQOT_服务

服务

权限

类型 :MQLONG-输入

权限。

对于 *Filter* 参数, 将忽略此字段。

对于 *AuthorityBuffer* 参数, 此字段表示实体对 *ProfileName* 和 *ObjectType* 标识的对象的权限。如果实体只有一个权限, 那么该字段等于相应的授权值 (MQZAO_* 常量)。如果实体具有多个权限, 那么该字段是相应 MQZAO_* 常量的按位 OR。

EntityDataPtr

类型 :PMQZED-输入

标识实体的 MQZED 结构的地址。

对于 过滤器 参数, 此字段指向 MQZED 结构, 该结构标识需要权限数据的实体。如果 *EntityDataPtr* 是空指针, 那么将返回所有实体的权限数据。

对于 *AuthorityBuffer* 参数, 此字段指向用于标识已返回权限数据的实体的 MQZED 结构。

EntityType

类型 :MQLONG-输入

实体类型。

对于 过滤器 参数, 此字段指定需要权限数据的实体类型。如果值为 MQZAET_NONE, 那么将返回所有实体类型的权限数据。

对于 *AuthorityBuffer* 参数, 此字段指定由 *EntityDataPtr* 参数指向的 MQZED 结构标识的实体类型。

值为下列其中一项; 对于 *Filter* 参数, 值 MQZAET_NONE 也有效:

MQZAET_PRINCIPAL

主体

MQZAET_GROUP

组

选项

类型:MQAUTHOPT-输入

选项。此字段指定用于控制所显示的概要文件的选项。必须指定下列其中一个值:

MQAUTHOPT_NAME_ALL_MATCHING

显示所有概要文件

MQAUTHOPT_NAME_EXPLICIT

显示与 *ProfileName* 字段中指定的名称完全相同的概要文件。

此外,还必须指定下列其中一项:

MQAUTHOPT_ENTITY_SET

显示用于计算实体对 *ProfileName* 参数指定的对象具有的累积权限的所有概要文件。*ProfileName* 参数不得包含任何通配符。

如果指定的实体是主体,那么对于集合 {entity, groups} 的每个成员,将显示适用于该对象的最适用概要文件。

如果指定的实体是组,那么将显示适用于该对象的组中最适用的概要文件。

如果指定了此值,那么 *ProfileName*, *ObjectType*, *EntityType* 的值以及在 *EntityDataPtr* MQZED 结构中指定的实体名称都必须为非空白。

如果指定了 MQAUTHOPT_NAME_ALL_MATCHING,那么还可以指定以下值:

MQAUTHOPT_ENTITY_EXPLICIT

显示与 *EntityDataPtr* MQZED 结构中指定的实体名称完全相同的概要文件。

C 声明

```
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 */
};
```

字段

MQZED-实体描述符

MQZED 结构用于许多授权服务调用,以指定要检查其授权的实体。

表 1. 汇总了结构中的字段。

| 字段 | 描述 |
|-----------------------|------|
| <u>StrucId</u> | 结构标识 |
| <u>版本</u> | 版本 |
| <u>EntityName Ptr</u> | 实体名称 |

表 598: MQZED 中的字段 (继续)

| 字段 | 描述 |
|------------------------|-------|
| <u>EntityDomainPtr</u> | 实体域指针 |
| <u>SecurityId</u> | 安全性标识 |
| <u>CorrelationPtr</u> | 关联指针 |

字段

StrucId

类型: MQCHAR4 -输入

结构标识。值如下所示:

MQZED_STRUC_ID

实体描述符结构的标识。

对于 C 编程语言, 还定义了常量 MQZED_STRUC_ID_ARRAY; 此值与 MQZED_STRUC_ID 相同, 但是字符数组而不是字符串。

版本

类型 :MQLONG-输入

结构版本号。值如下所示:

MQZED_VERSION_1

Version-1 实体描述符结构。

以下常量指定当前版本的版本号:

MQZED_CURRENT_VERSION

实体描述符结构的当前版本。

EntityNamePtr

类型 :PMQCHAR-输入

概要文件名称。

实体名称的地址。这是指向要检查其授权的实体的名称的指针。

EntityDomainPtr

类型 :PMQCHAR-输入

实体域名的地址。这是一个指向域名称的指针, 该域包含要检查其授权的实体的定义。

SecurityId

类型: MQBYTE40 -输入

权限。

安全标识。这是要检查其权限的安全标识。

CorrelationPtr

类型 :MQPTR-输入

关联指针。这有助于在认证用户函数与其他相应的 OAM 函数之间传递相关数据。

C 声明

```
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 */
```

字段

MQZEP-添加组件入口点

服务组件在初始化期间启动此函数，以将入口点添加到该服务组件的入口点向量。

语法

```
MQZEP (Hconfig, Function, EntryPoint, CompCode, Reason)
```

参数

Hconfig

类型:MQHCONFIG-输入

配置句柄。此句柄表示正在为此特定可安装服务配置的组件。它必须与队列管理器在组件初始化调用时传递到组件配置函数的组件相同。

函数

类型:MQLONG-输入

函数标识。为此，将为每个可安装服务定义有效值。

如果针对同一函数多次调用 MQZEP，那么最后一次调用将提供所使用的入口点。

EntryPoint

类型:PMQFUNC-输入

函数入口点。这是组件为执行该功能而提供的入口点的地址。

值 NULL 有效，指示此组件未提供该函数。对于未使用 MQZEP 定义的入口点，假定为 NULL。

CompCode

类型:MQLONG - 输出

完成代码。它必须是下列其中一个值:

MQCC_OK

成功完成。

MQCC_FAILED

调用失败。

原因

类型:MQLONG - 输出

原因码限定 *CompCode*。

如果 *CompCode* 为 MQCC_OK:

MQRC_NONE

(0, X'000') 没有要报告的原因。

如果 *CompCode* 为 MQCC_FAILED:

MQRC_FUNCTION_ERROR

(2281, X'8E9') 函数标识无效。

MQRC_HCONFIG_ERROR

(2280, X'8E8') 配置句柄无效。

有关这些原因码的更多信息，请参阅 [API 原因码](#)。

C 调用

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

按如下所示声明参数:

```
MQHCONFIG Hconfig; /* Configuration handle */
MQLONG Function; /* Function identifier */
PMQFUNC EntryPoint; /* Function entry point */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Reason code qualifying CompCode */
```

MQZFP-可用参数

MQZFP 结构用于 *FreeParms* 参数的 MQZ_FREE_USER 调用。此参数指定与要释放的资源相关的数据。

表 1。汇总了结构中的字段。

| 表 599: MQZFP 中的字段 | |
|--------------------------------|------|
| 字段 | 描述 |
| StrucId | 结构标识 |
| 版本 | 版本 |
| 已保留 | 保留字段 |
| CorrelationPtr | 关联指针 |

字段

StrucId

类型: MQCHAR4 -输入

结构标识。值如下所示:

MQZIC_STRUC_ID

身份上下文结构的标识。对于 C 编程语言,还定义了常量 MQZIC_STRUC_ID_ARRAY;此值与 MQZIC_STRUC_ID 相同,但是字符数组而不是字符串。

版本

类型:MQLONG-输入

结构版本号。值如下所示:

MQZFP_VERSION_1

Version-1 可用参数结构。

以下常量指定当前版本的版本号:

MQZFP_CURRENT_VERSION

当前版本的免费参数结构。

已保留

类型: MQBYTE8 -输入

保留字段。初始值为空。

CorrelationPtr

类型: MQPTR-输入

关联指针。与要释放的资源相关的关联数据的地址。

C 声明

```
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 */
};
```

字段

MQZIC-身份上下文

MQZIC 结构用于 *IdentityContext* 参数的 MQZ_AUTHENTICATE_USER 调用。

MQZIC 结构包含身份上下文信息，用于标识首先将消息放入队列的应用程序的用户：

- 队列管理器使用标识用户的名称填充 *UserIdentifier* 字段，队列管理器执行此操作的方式取决于运行应用程序的环境。
- 队列管理器在 *AccountingToken* 字段中填充从放置消息的应用程序中确定的令牌或数字。
- 应用程序可以使用 *ApplIdentity* 数据 字段来获取他们想要包含的有关用户的任何额外信息 (例如，加密密码)。

适当授权的应用程序可以使用 MQZ_AUTHENTICATE_USER 函数来设置身份上下文。

在 WebSphere MQ for Windows 下创建消息时，Windows 系统安全标识 (SID) 存储在 *AccountingToken* 字段中。SID 可用于补充 *UserIdentifier* 字段和建立用户的凭证。

表 1。 汇总了结构中的字段。

| 字段 | 描述 |
|----------------------------------|----------|
| StrucId | 结构标识 |
| 版本 | 版本 |
| UserIdentifier | 用户标识 |
| AccountingToken | 记帐标记 |
| ApplIdentityData | 应用程序标识数据 |

字段

StrucId

类型: MQCHAR4 -输入

结构标识。 值如下所示:

MQZIC_STRUC_ID

身份上下文结构的标识。 对于 C 编程语言，还定义了常量 MQZIC_STRUC_ID_ARRAY; 此值与 MQZIC_STRUC_ID 相同，但是字符数组而不是字符串。

版本

类型 :MQLONG-输入

结构版本号。 值如下所示:

MQZIC_VERSION_1

Version-1 身份上下文结构。

以下常量指定当前版本的版本号:

MQZIC_CURRENT_VERSION

当前版本的身份上下文结构。

UserIdentifier

类型: MQCHAR12 -输入

用户标识。这是消息的身份上下文的一部分。*UserIdentifier* 指定发出消息的应用程序的用户标识。队列管理器将此信息视为字符数据，但不定义其格式。有关 *UserIdentifier* 字段的更多信息，请参阅第 401 页的『[UserIdentifier \(MQCHAR12\)](#)』。

AccountingToken

类型: MQBYTE32 -输入

记帐标记。这是消息的身份上下文的一部分。*AccountingToken* 允许应用程序将由于消息而完成的工作相应地收费。队列管理器将此信息视为位字符串，并且不检查其内容。有关 *AccountingToken* 字段的更多信息，请参阅第 367 页的『[AccountingToken \(MQBYTE32\)](#)』。

ApplIdentityData

类型: MQCHAR32 -输入

与身份相关的应用程序数据。这是消息的身份上下文的一部分。*ApplIdentity* 数据是应用程序套件定义的信息，可用于提供有关消息源的其他信息。例如，它可以由使用适当用户权限运行的应用程序设置，以指示身份数据是否可信。有关 *ApplIdentity* 数据字段的更多信息，请参阅第 369 页的『[ApplIdentity 数据 \(MQCHAR32\)](#)』。

C 声明

```
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 */
};
```

字段

IBM WebSphere MQ Bridge for HTTP 的参考材料

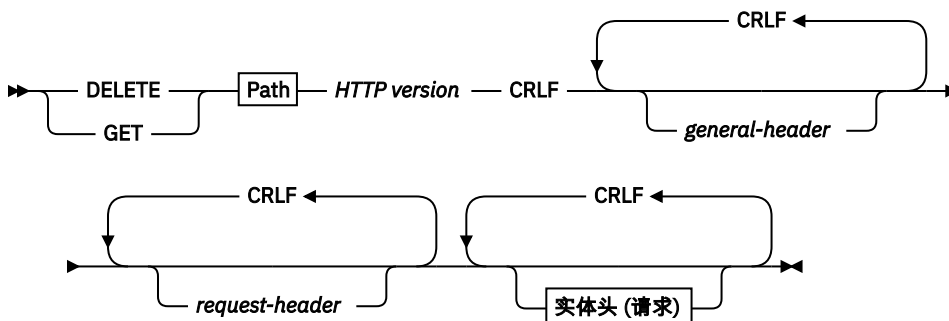
IBM WebSphere MQ Bridge for HTTP 的参考主题，按字母顺序排列

HTTP DELETE: WebSphere MQ bridge for HTTP 命令

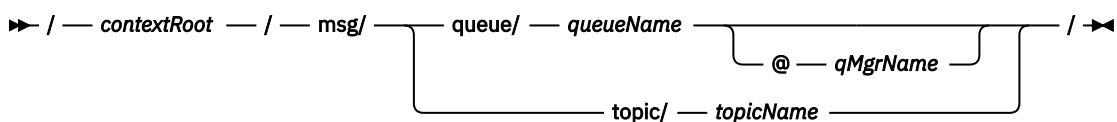
HTTP **DELETE** 操作从 WebSphere MQ 队列获取消息，或者从主题检索发布内容。将从队列中除去该消息。如果保留发布，那么不会将其除去。响应消息将发送回客户机，包括有关该消息的信息。

语法

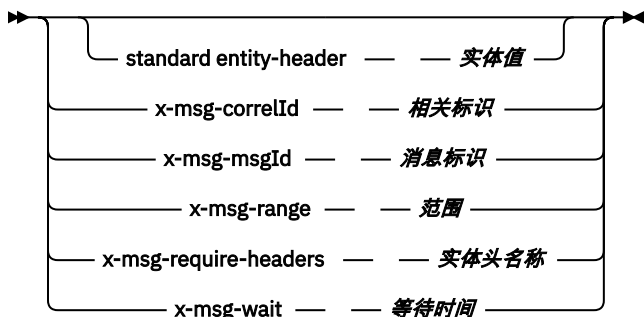
请求



Path



entity-header (请求)

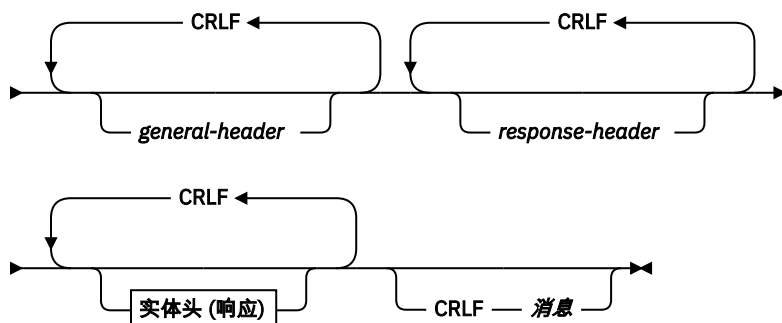


注:

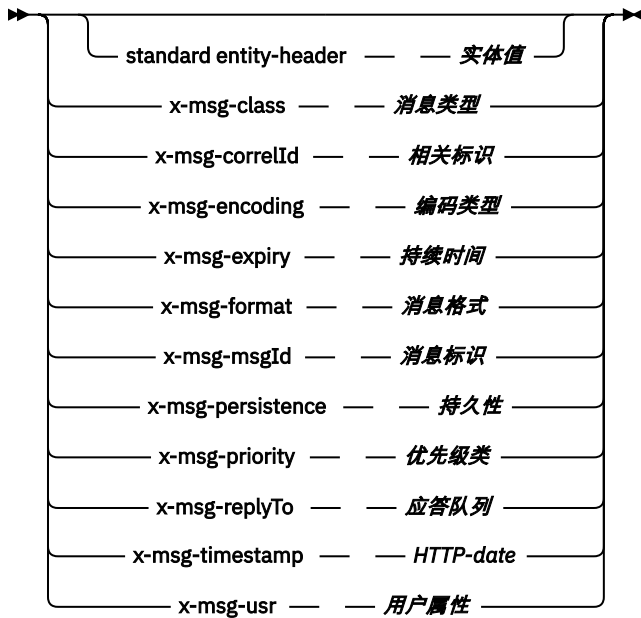
1. If a question mark (?) is used it must be substituted with %3f. For example, orange?topic should be specified as orange%3ftopic.
2. `@qMgrName` is only valid on an HTTP **POST**

响应

►► `HTTP version` — `HTTP Status-Code` — `HTTP Reason-Phrase` — CRLF ►►



entity-header (响应)



请求 参数

路径

请参阅 [第 1119 页的『URI 格式』](#)。

HTTP 版本

HTTP 版本; 例如, HTTP/1.1

general-header

请参阅 [HTTP/1.1 - 4.5 常规头字段](#)。

请求头

请参阅 [HTTP/1.1 - 5.3 请求头字段](#)。Host 字段是 HTTP/1.1 请求的必填字段。它通常由用于创建客户机请求的工具自动插入。

entity-header (请求)

请参阅 [HTTP/1.1 - 7.1 实体头字段](#)。请求语法图中列出的其中一个实体头。

响应 参数

路径

请参阅 [第 1119 页的『URI 格式』](#)。

HTTP 版本

HTTP 版本; 例如, HTTP/1.1

general-header

请参阅 [HTTP/1.1 - 4.5 常规头字段](#)。

response-header

请参阅 [HTTP/1.1 - 6.2 响应头字段](#)。

entity-header (响应)

请参阅 [HTTP/1.1 - 7.1 实体头字段](#)。"响应" 语法图中列出的其中一个实体或响应头。Content-Length 始终存在于响应中。如果没有消息体, 那么将其设置为零。

message

消息体。

描述

如果 HTTP **DELETE** 请求成功，那么响应消息将包含从 WebSphere MQ 队列检索到的数据。在 HTTP Content-Length 头中返回消息体中的字节数。HTTP 响应的状态码设置为 200 OK。如果将 x-msg-range 指定为 0 或 0-0，那么 HTTP 响应的状态码为 204 No Content。

如果 HTTP **DELETE** 请求失败，那么响应包含用于 HTTP 错误消息的 WebSphere MQ 网桥和 HTTP 状态码。

HTTP DELETE 示例

HTTP **DELETE** 从队列中获取并删除消息，或者检索并删除发布。**HTTPDELETE** Java 样本是从队列中读取消息的 HTTP **DELETE** 请求示例。不使用 Java，您可以改用浏览器表单或 AJAX 工具包创建 HTTP **DELETE** 请求。

第 1090 页的图 37 是用于删除队列中名为 myQueue 的下一条消息的 HTTP 请求。作为响应，消息体将返回给客户机。在 WebSphere MQ 术语中，HTTP **DELETE** 是破坏性获取。

该请求包含 HTTP 请求头 x-msg-wait，用于指示 WebSphere MQ Bridge for HTTP 等待消息到达队列的时间长度。此请求还包含 x-msg-require-headers 请求头，用于指定客户机将在响应中收到消息相关标识。

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

图 37: HTTP **DELETE** 请求示例

第 1090 页的图 38 是返回给客户机的响应。相关标识将返回至客户机，如请求的 x-msg-require-headers 中所请求。

```
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.
```

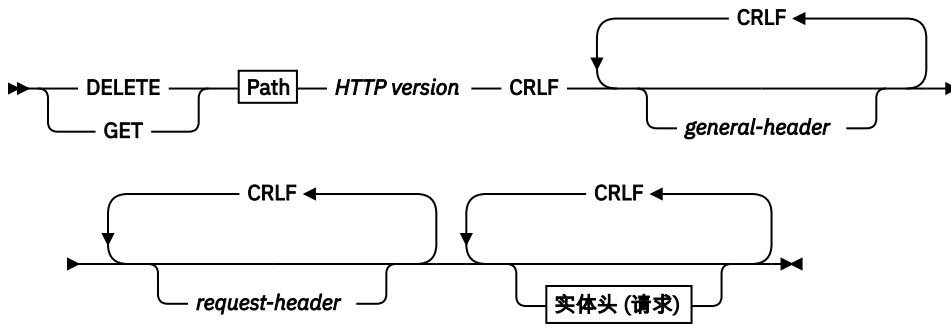
图 38: HTTP **DELETE** 响应示例

HTTP GET: WebSphere MQ bridge for HTTP 命令

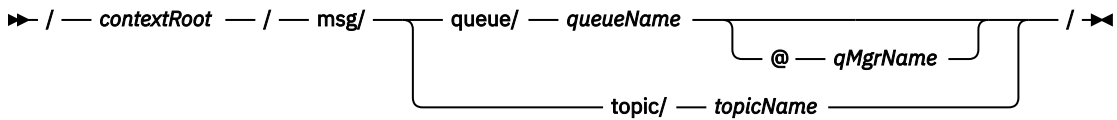
HTTP **GET** 操作从 WebSphere MQ 队列获取消息。此消息将保留在队列中。HTTP **GET** 操作相当于浏览 WebSphere MQ 队列。

语法

请求



Path



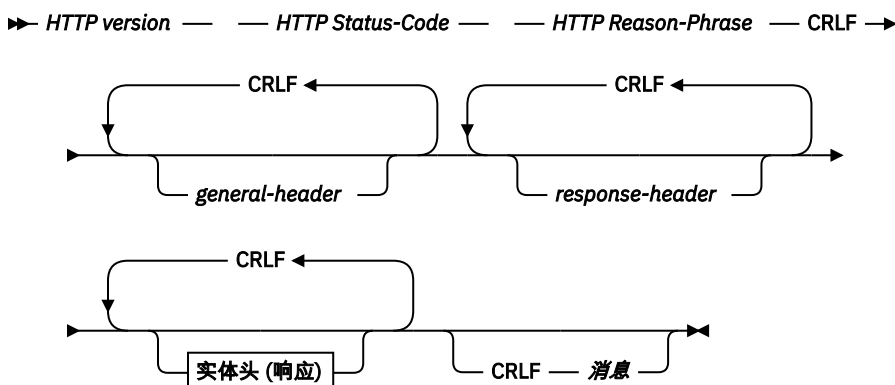
entity-header (请求)



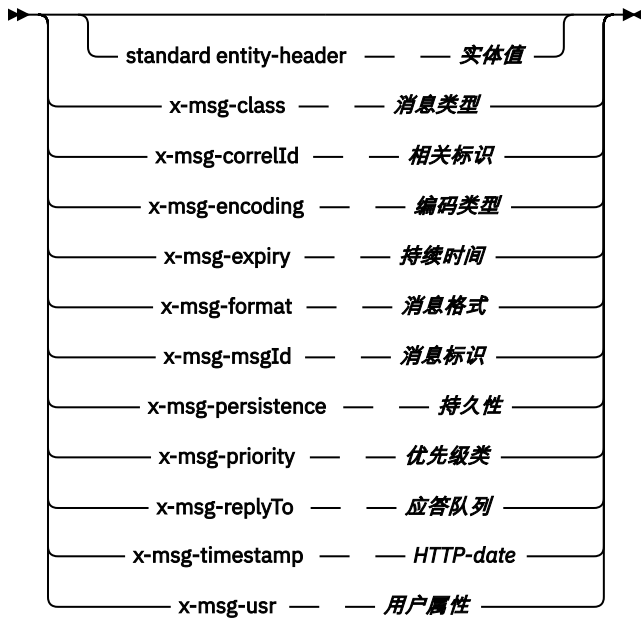
注:

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**

响应



entity-header (响应)



请求 参数

路径

请参阅 [第 1119 页的『URI 格式』](#)。

HTTP 版本

HTTP 版本; 例如, HTTP/1.1

general-header

请参阅 [HTTP/1.1 - 4.5 常规头字段](#)。

请求头

请参阅 [HTTP/1.1 - 5.3 请求头字段](#)。Host 字段是 HTTP/1.1 请求的必填字段。它通常由用于创建客户机请求的工具自动插入。

entity-header (请求)

请参阅 [HTTP/1.1 - 7.1 实体头字段](#)。请求语法图中列出的其中一个实体头。

响应 参数

路径

请参阅 [第 1119 页的『URI 格式』](#)。

HTTP 版本

HTTP 版本; 例如, HTTP/1.1

general-header

请参阅 [HTTP/1.1 - 4.5 常规头字段](#)。

response-header

请参阅 [HTTP/1.1 - 6.2 响应头字段](#)。

entity-header (响应)

请参阅 [HTTP/1.1 - 7.1 实体头字段](#)。"响应"语法图中列出的其中一个实体或响应头。Content-Length 始终存在于响应中。如果没有消息体, 那么将其设置为零。

message

消息体。

描述

如果 HTTP **GET** 请求成功，那么响应消息将包含从 WebSphere MQ 队列检索到的数据。在 HTTP Content-Length 头中返回消息体中的字节数。HTTP 响应的状态码设置为 200 OK。如果将 x-msg-range 指定为 0 或 0-0，那么 HTTP 响应的状态码为 204 No Content。

如果 HTTP **GET** 请求失败，那么响应包含用于 HTTP 错误消息的 WebSphere MQ 网桥和 HTTP 状态码。

HTTP GET 示例

HTTP **GET** 从队列中获取消息。该消息保留在队列中。在 WebSphere MQ 术语中，HTTP **GET** 是浏览请求。您可以使用 Java 客户机、浏览器表单或 AJAX 工具包创建 HTTP **GET** 请求。

第 1093 页的图 39 是用于浏览名为 myQueue 的队列上的下一条消息的 HTTP 请求。

该请求包含 HTTP 请求头 x-msg-wait，用于指示 WebSphere MQ Bridge for HTTP 等待消息到达队列的时间长度。此请求还包含 x-msg-require-headers 请求头，用于指定客户机将在响应中收到消息相关标识。

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

图 39: HTTP **GET** 请求示例

第 1093 页的图 40 是返回到客户机的响应。相关标识将返回至客户机，如请求的 x-msg-require-headers 中所请求。

```
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.
```

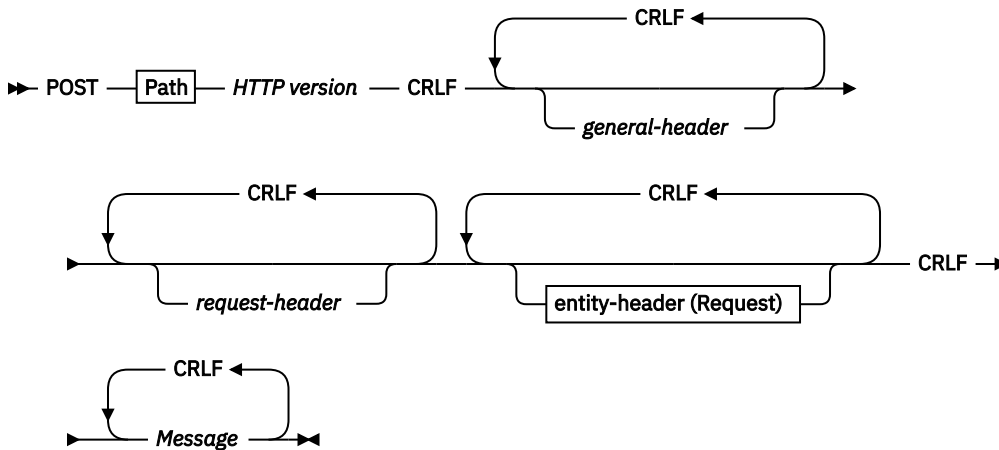
图 40: HTTP **GET** 响应示例

HTTP POST: WebSphere MQ bridge for HTTP 命令

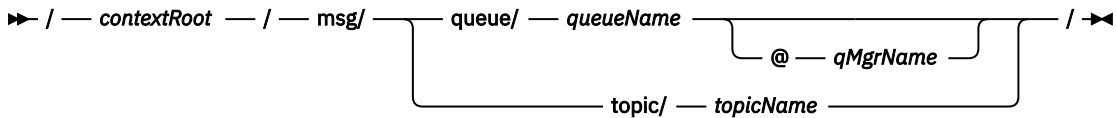
HTTP **POST** 操作将消息放在 WebSphere MQ 队列上，或者将消息发布到主题。

Syntax

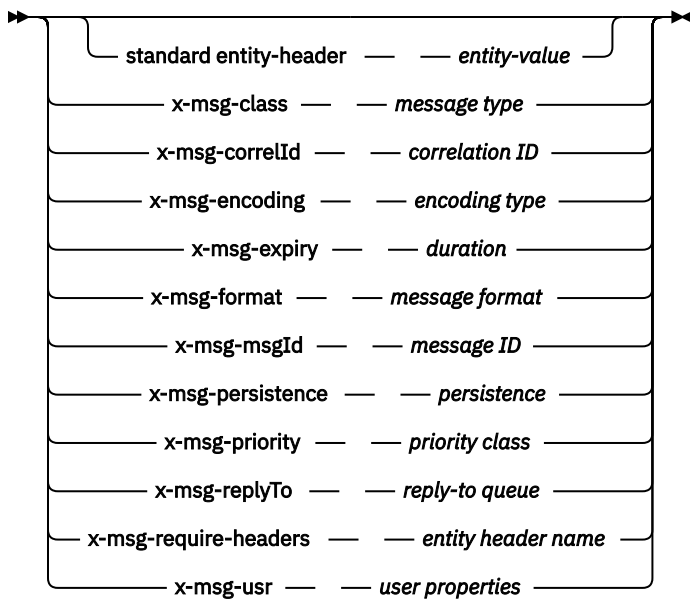
Request



Path



entity-header (Request)

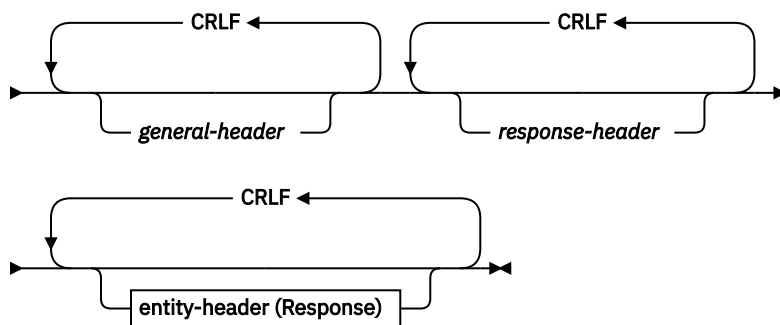


注:

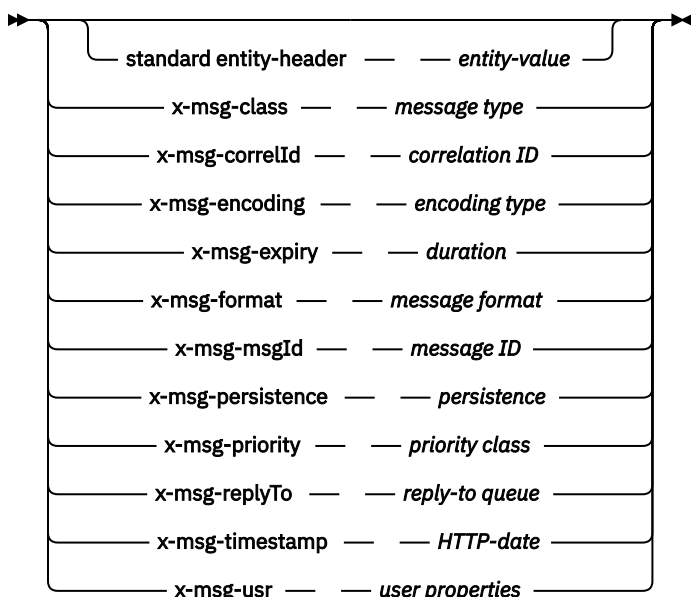
1. If a question mark (?) is used it must be substituted with %3f. For example, orange?topic should be specified as orange%3ftopic.
2. @qMgrName is only valid on an HTTP **POST**

Response

→ HTTP version — — HTTP Status-Code — — HTTP Reason-Phrase — CRLF →



entity-header (Response)



请求参数

路径

请参阅第 1119 页的『URI 格式』。

HTTP 版本

HTTP 版本; 例如, HTTP/1.1

general-header

请参阅 [HTTP/1.1 - 4.5 常规头字段](#)。

请求头

请参阅 [HTTP/1.1 - 5.3 请求头字段](#)。Host 字段是 HTTP/1.1 请求的必填字段。它通常由用于创建客户机请求的工具自动插入。

entity-header (请求)

请参阅 [HTTP/1.1 - 7.1 实体头字段](#)。请求语法图中列出的其中一个实体头。Content-Length 和 Content-Type 应插入到请求中, 并且通常由用于创建客户机请求的工具自动插入。Content-Type 必须与 x-msg-class 定制 entity-header (如果已指定) 中定义的类型匹配。

message

要放入队列的消息, 或要发布到主题的发布。

响应 参数

路径

请参阅 [第 1119 页的『URI 格式』](#)。

HTTP 版本

HTTP 版本; 例如, HTTP/1.1

general-header

请参阅 [HTTP/1.1 - 4.5 常规头字段](#)。

response-header

请参阅 [HTTP/1.1 - 6.2 响应头字段](#)。

entity-header (响应)

请参阅 [HTTP/1.1 - 7.1 实体头字段](#)。"响应" 语法图中列出的其中一个实体或响应头。Content-Length 始终存在于响应中。如果没有消息体, 那么将其设置为零。

描述

如果未包含 x-msg-usr 头, 并且消息类为 BYTES 或 TEXT, 那么放入队列中的消息没有 MQRFH2。

使用 HTTP **POST** 请求中的 HTTP 实体和请求头来设置放入队列中的消息的属性。您还可以使用 x-msg-require-headers 来请求在响应消息中返回哪些头。

如果 HTTP **POST** 请求成功, 那么响应消息的实体为空, 其内容长度为零。HTTP 状态码为 200 OK。

如果 HTTP **POST** 请求失败, 那么响应包含用于 HTTP 错误消息的 WebSphere MQ 网桥和 HTTP 状态码。不会将 WebSphere MQ 消息放入队列或主题中。

HTTP POST 示例

HTTP **POST** 将消息放入队列, 或将发布内容放入到主题中。HTTP**POST** Java 样本是将消息放入队列的 HTTP **POST** 请求示例。您可以使用浏览器表单或 AJAX 工具箱来创建 HTTP **POST** 请求, 而不使用 Java。

[第 1096 页的图 41](#) 显示了用于将消息放入名为 myQueue 的队列的 HTTP 请求。此请求包含 HTTP 头 x-msg-correlID, 用于设置 WebSphere MQ 消息的相关标识。

```
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.
```

图 41: 对队列的 HTTP **POST** 请求的示例

[第 1096 页的图 42](#) 显示了发送回客户机的响应。没有任何响应内容。

```
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
```

图 42: HTTP **POST** 响应示例

HTTP 头

WebSphere MQ Bridge for HTTP 支持定制请求 HTTP 头, 定制实体 HTTP 头和部分标准 HTTP 头。

HTTP 实践是使用 x- 作为所有定制头的前缀, WebSphere MQ Bridge for HTTP 头以 x-msg- 作为前缀。例如, 要设置优先级头, 请使用 x-msg-priority。

注:

- 大多数头值区分大小写。例如, 使用 `msgId` 头时, `NONE` 是关键字, 而 `none` 是 `msgID`。
- 将忽略拼写错误的头。

定制实体 HTTP 头

定制实体 HTTP 头包含有关 WebSphere MQ 消息的信息。通过使用实体头, 可以在消息描述符 (MQMD) 中设置值, 或在 MQMD 中设置查询值。其他实体头 `x-msg-usr` 设置并返回要与请求关联的任何用户属性信息。

可以在不同的 HTTP 请求上下文中使用实体头:

DELETE

只能将 `x-msg-correlId` 和/或 `x-msg-msgId` 实体头与 **DELETE** HTTP 请求配合使用。头的作用是通过 MQGET 中的 `MsgId` 和 `CorrelId` 选择特定消息, 并从其队列中删除该消息。

GET

只能将 `x-msg-correlId` 和/或 `x-msg-msgId` 实体头与 **GET** HTTP 请求配合使用。头的作用是通过 `MsgId` 和 `CorrelId` 在 MQGET 中选择特定消息以进行浏览。

POST

您可以在 **POST** HTTP 请求中使用任何实体头, 但 `x-msg-timestamp` 除外。

x-msg-require-headers

在任何 **GET**, **POST** 或 **DELETE** HTTP 请求上, 可以在 `x-msg-require-headers` 请求头中添加多个实体头 (以逗号分隔)。结果是返回 HTTP 响应消息中的指定实体头, 其中包含关联消息属性的值。

每个头列表的描述, WebSphere MQ Bridge for HTTP 在其中处理头的上下文。例如, 在头 **POST**, `x-msg-require-headers` 中, 头由 HTTP **POST** 请求中的 WebSphere MQ Bridge for HTTP 处理, 或者由 HTTP **POST**, **GET** 或 **DELETE** 请求中的 `x-msg-require-headers` 请求头处理。如果头包含在不允许该头的上下文中, 那么将忽略该头。未报告任何错误。

您可以将任何标准 HTTP 头放入要由 Web 服务器或其他请求处理程序处理的请求中。同样, 响应可能包含由 Web 服务器或其他响应处理程序插入的其他标准 HTTP 头。

定制请求 HTTP 头

三个定制请求 HTTP 头 `x-msg-range`, `x-msg-require-headers` 和 `x-msg-wait` 将有关 HTTP 请求的其他信息传递到服务器。它们充当请求修饰符。通过 `x-msg-range`, 您可以限制在响应中返回的消息数据量。通过 `x-msg-require-headers`, 您可以请求响应以包含有关请求结果的信息。使用 `x-msg-wait`, 可以修改客户机等待 HTTP 响应的的时间。

标准 HTTP 头

必须在 HTTP/1.1 请求中指定 Host 标准 HTTP request-header。

可以在请求中指定 Content-Length 和 Content-Type 标准 HTTP 实体头。

可以返回 Content-Length, Content-Location, Content-Range, Content-Type 和 Server 标准 HTTP 实体头以响应请求。在请求消息的 `x-msg-request-header` 头中指定一个或多个标准 HTTP 头。

HTTP 头的字母列表

class:HTTP x-msg-class entity-header

设置或返回消息类型。

| 类型 | 描述 |
|----------|--------------------------|
| HTTP 头名称 | <code>x-msg-class</code> |
| HTTP 头类型 | 实体标题 |

| 类型 | 描述 |
|----------------|--|
| 在 HTTP 请求消息中有效 | POST, x-msg-require-headers |
| 允许的值 | BYTES MAP STREAM TEXT |
| 缺省值 | BYTES |

描述

- 在 HTTP **POST** 请求中，设置所创建消息的类型。
- 在 **GET** 或 **DELETE** 上指定类头将返回实体主体为 MQHTTP40007 的 400 Bad Request。
- 在 x-msg-require-headers 中指定，将 HTTP 响应消息中的 x-msg-class 设置为消息类型。
- 如果为此头指定了无效值，那么将返回 MQHTTP40005 消息。
- 如果未指定 x-msg-class 头，并且消息的内容类型为 application/x-www-form-urlencoded，那么假定数据是 JMS 映射对象。

Content-Length:HTTP 实体头

设置或返回消息主体的长度 (以字节计)。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | 内容长度 |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | x-msg-require-headers |
| 允许的值和返回的值 | Integer value 消息体的长度 (以字节为单位)。 |

描述

- Content-Length 在 HTTP 请求中是可选的。对于 **GET** 或 **DELETE**，长度必须为零。对于 **POST**，如果指定了 Content-Length 并且它与消息行的长度不匹配，那么消息将被截断，或者用空值填充到指定的长度。
- Content-Length 始终在 HTTP 响应中返回，即使没有内容时也是如此，在这种情况下，值为零。

Content-Location:HTTP 实体头

在 HTTP 响应消息的标准 Content-Location 头中返回请求中引用的队列或主题。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | 内容-位置 |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | x-msg-require-headers |
| 返回的值 | 格式为 URI， <code>/msg/queue/queuename</code> 或 |

| 类型 | 描述 |
|----|-----------------------------------|
| | <code>/msg/topic/topicname</code> |

描述

- 在 `x-msg-require-headers` 中请求时，Content-Location 实体头将返回 HTTP 请求中引用的队列或主题。

Content-Range:HTTP entity-header

返回从 HTTP 响应的 Content-Range 头中的 WebSphere MQ 消息中选择的字节范围。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | 内容范围 |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | <code>x-msg-require-headers</code> |
| 返回的值 | <p>String 返回返回的子串的下限 <i>m</i> 和上限 <i>n</i> 以及整个消息的 <i>length</i>。例如，</p> <p><code>m-n/length</code></p> |

描述

- 仅当在包含 `x-msg-range` 请求头的 **GET** 或 **DELETE** 请求中指定了 Content-Range 时，才会在 HTTP 响应中返回 Content-Range。
 - 如果在 **GET** 或 **DELETE** 请求上指定了 `x-msg-range`，那么将在响应中返回 Content-Range 头中指定的字节范围。例如，如果在包含 100 字节的消息的请求中使用 `x-msg-range: 0-60`，那么 `content-range` 头将保存字符串 `0-60/100`
 - `x-msg-range` 请求还会返回 HTTP 响应中 `x-msg-range` 头中的内容范围。

Content-Type:HTTP 实体头

根据 HTTP content-type 设置或返回 WebSphere MQ 消息中 JMS 消息的类。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | 内容类型 |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , <code>x-msg-require-headers</code> |
| 允许的值或返回的值 | <p>media-type 对于受支持的介质类型，请参阅第 1099 页的表 601。</p> |

表 601: `x-msg-class` 与 HTTP Content-Type 之间的映射

| <code>x-msg-class</code> | HTTP 内容类型 |
|--------------------------|---|
| BYTES | <p><code>application/octet-stream</code> <code>application/xml</code></p> |

表 601: *x-msg-class* 与 *HTTP Content-Type* 之间的映射 (继续)

| x-msg-class | HTTP 内容类型 |
|--------------------|---|
| TEXT | text/* |
| MAP | application/x-www-form-urlencoded application/xml (可选) |
| STREAM | application/xml (可选) |

描述

- 在 HTTP **POST** 请求上, 指定 *Content-Type* 或 *x-msg-class*。如果同时指定这两者, 那么它们必须一致, 否则将返回 HTTP Bad Request 异常 Status code 400。如果同时省略 *Content-Type* 和 *x-msg-class*, 那么假定 *Content-Type* 为 text/*。
- Content-Type* 始终设置在对具有消息体的 HTTP **GET** 或 **DELETE** 的响应中。 *Content-Type* 是根据第 1100 页的表 602 中的规则设置的。

表 602: 将消息类型映射到 *x-msg-class* 和 *Content-Type*

| 消息格式 | JMS 消息类型 | <i>x-msg-class</i> | 内容类型 |
|------------------------|------------|--------------------|--------------------------|
| 除 MQFMT_STRING 以外的任何内容 | None | BYTES | application/octet-stream |
| MQFMT_STRING | None | TEXT | text/plain |
| MQFMT_NONE | jms_bytes | BYTES | application/octet-stream |
| MQFMT_NONE | jms_text | TEXT | text/plain |
| MQFMT_NONE | jms_map | MAP | application/xml |
| MQFMT_NONE | jms_stream | STREAM | application/xml |

correlId:HTTP x-msg-correlId entity-header

设置或返回相关标识。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | x-msg-correlId |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | DELETE, GET, POST, x-msg-require-headers |
| 允许的值 | <p>String value</p> <p>例如:</p> <pre>x-msg-correlId: mycorrelationid</pre> <p>允许使用引号括起的字符串; 例如:</p> <pre>x-msg-correlId: "my id"</pre> |

| 类型 | 描述 |
|-----|---|
| | <p>Hex value 以 0x: 为前缀的十六进制值; 例如:</p> <pre>x-msg-correlId: 0x:43c1d23a</pre> <p>0x: 后面的十六进制值限制为 48 个字符, 表示 24 个字节。将忽略其他数据。</p> |
| 缺省值 | 不适用 |

描述

- 在 HTTP **POST** 请求上, 设置所创建消息的相关标识。
- 在 HTTP **GET** 或 **DELETE** 请求上, 从队列或主题中选择消息。如果不存在具有指定相关标识的消息, 那么将返回 HTTP 504 Gateway Timeout 响应。x-msg-correlId 可与 x-msg-msgID 配合使用, 以从队列或主题中选择与两个选择器都匹配的消息。
- 在 x-msg-require-headers 中指定, 将 HTTP 响应消息中的 x-msg-coreId 设置为消息的相关标识。
- 在 0x: 前缀后允许使用水平空格。

注:

- 在 HTTP **GET** 或 **DELETE** 请求上指定不带值的 x-msg-correlId; 例如, "x-msg-correlId:", 返回队列或主题上的下一条消息, 而不考虑其相关标识。
- 如果指定 24 个字符或更少的选择器, 或者指定 0x: 后跟 48 个字符或更少的选择器, 那么 WebSphere MQ Bridge for HTTP 将使用优化的选择器来提高性能。
- 从队列中选择消息时, 将使用包含 JMSCorrelationID 的 JMS 消息选择器。此选择器的行为如 [选择行为中](#) 所述。

encoding:HTTP x-msg-encoding entity-header

设置或返回消息编码。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | x-msg-encoding |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , x-msg-require-headers |
| 允许的值 | <p>以下值的逗号分隔列表:</p> <p>DECIMAL_NORMAL DECIMAL_REVERSED FLOAT_IEEE_NORMAL FLOAT_IEEE_REVERSED FLOAT_S390 INTEGER_NORMAL INTEGER_REVERSED</p> <p>例如,</p> <pre>x-msg-encoding: INTEGER_NORMAL,DECIMAL_NORMAL,FLOAT_IEEE_NORMAL</pre> <p>注: 值区分大小写</p> |
| 缺省值 | DECIMAL_NORMAL, FLOAT_IEEE_NORMAL, INTEGER_NORMAL |

描述

- 在 HTTP **POST** 请求上，指定所创建消息的编码。
- 在 HTTP **GET** 或 **DELETE** 请求上，将忽略 `x-msg-encoding` 头。
- 在 `x-msg-require-headers` 中指定，将 HTTP 响应消息中的 `x-msg-encoding` 设置为消息的编码属性。

到期:HTTP `x-msg-到期` entity-header

设置或返回消息到期持续时间。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | <code>x-msg-到期</code> |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , <code>x-msg-require-headers</code> |
| 允许的值 | UNLIMITED 例如; <pre>x-msg-expiry: UNLIMITED</pre> Integer value 到期前的毫秒数。 例如; <pre>x-msg-expiry: 10000</pre> |
| 缺省值 | UNLIMITED |

描述

- 在 HTTP **POST** 请求上设置时，请求消息将在指定的时间内到期。
- 在 HTTP **GET** 或 **DELETE** 请求上，将忽略 `x-msg-到期` 头。
- 在 `x-msg-require-headers` 中指定，将 HTTP 响应消息中的 `x-msg-到期` 设置为消息的到期时间。
- UNLIMITED 指定消息永不到期。
- 消息的到期从消息到达队列时开始，因此将忽略网络等待时间。
- 最大值由 WebSphere MQ 限制为 214748364700 毫秒。如果指定了大于此值的值，那么将采用最大可能的到期时间。

format:HTTP `x-msg-format` entity-header

设置或返回 WebSphere MQ 消息格式。

| 类型 | 描述 |
|----------------|---|
| HTTP 头名称 | <code>x-msg-format</code> |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , <code>x-msg-require-headers</code> |
| 允许的值 | NONE 例如, <pre>x-msg-format: NONE</pre> |

| | |
|-----|--|
| 类型 | 描述 |
| | <p>String value 用户定义的任何值 (最多 8 个字符)。例如,</p> <pre>x-msg-format: myformat</pre> |
| 缺省值 | None |

描述

- 在 HTTP **POST** 请求上设置时, 请设置请求消息格式。
- 在 HTTP **GET** 或 **DELETE** 请求上, 将忽略 `x-msg-format` 头。
- 在 `x-msg-require-headers` 中指定, 将 HTTP 响应消息中的 `x-msg-format` 设置为消息的格式。
- **NONE** 区分大小写, 指示消息格式为空白。
- 将使用 `x-msg-format` 的值, 即使它与 HTTP 请求的介质类型相冲突也是如此。请参阅第 1103 页的表 603。

| x-msg-class | 内容类型 | 队列/主题上的消息格式 |
|--------------------|---|---|
| BYTES | <ul style="list-style-type: none"> • application/octet-stream • application/xml | WebSphere MQ 消息: MQFMT 设置为 MQC.MQFMT_NONE |
| TEXT | <ul style="list-style-type: none"> • text/* | WebSphere MQ 消息: MQFMT 设置为 MQC.MQFMT_STRING |
| MAP | <ul style="list-style-type: none"> • application/x-www-form-urlencoded • application/xml (可选) | JMSMap |
| STREAM | <ul style="list-style-type: none"> • application/xml (可选) | JMSStream |

msgId:HTTP x-msg-msgId entity-header

设置或返回消息标识。

| | |
|----------------|--|
| 类型 | 描述 |
| HTTP 头名称 | x-msg-msgId |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | DELETE, GET, POST, x-msg-require-headers |
| 允许的值 | <p>String value 例如,</p> <pre>x-msg-msgId: mymsgid</pre> <p>用引号括起的字符串, 例如, <code>x-msg-msgId: "my id"</code></p> <p>Hex value 以 <code>0x:</code> 为前缀的十六进制值; 例如,</p> <pre>x-msg-msgId: 0x:43c1d23a</pre> |

| 类型 | 描述 |
|-----|-----|
| 缺省值 | 不适用 |

描述

- 在 HTTP **POST** 请求上，设置所创建消息的消息标识。
- 在 HTTP **GET** 或 **DELETE** 请求上，从队列或主题中选择消息。如果不存在具有指定消息标识的消息，那么将返回 HTTP 504 Gateway Timeout 响应。x-msg-msgId 可与 x-msg-correlID 配合使用，以从与两个选择器都匹配的队列或主题中选择消息。
- 在 x-msg-require-headers 中指定，在对消息标识的 HTTP 响应中返回 x-msg-msgId。
- 在 0x: 前缀后允许使用水平空格。

注: 在 HTTP **GET** 或 **DELETE** 请求上指定 x-msg-msgId 而不指定值; 例如, "x-msg-msgId:" 将返回队列或主题上的下一条消息, 而不考虑其消息标识。

从队列中选择消息时, 将使用包含 JMSMessageID 的 JMS 消息选择器。此选择器的行为如 [选择行为](#) 中所述。

persistence:HTTP x-msg-persistence entity-header

设置或返回消息持久性。

| 类型 | 描述 |
|----------------|---|
| HTTP 头名称 | x-msg-持久性 |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , x-msg-require-headers |
| 允许的值 | <p>NON_PERSISTENT 此消息无法在系统故障或队列管理器重新启动后继续存在。 例如,</p> <pre>x-msg-persistence: NON_PERSISTENT</pre> <p>PERSISTENT 消息在队列管理器的系统故障和重新启动后仍然存在。 例如,</p> <pre>x-msg-persistence: PERSISTENT</pre> <p>AS_DESTINATION 仅适用于 POST。 使用由消息提供程序确定的目标的缺省持久性。</p> <p>注: 区分大小写</p> |
| 缺省值 | NON_PERSISTENT |

描述

- 在 HTTP **POST** 请求上设置时, 请设置请求消息持久性。
- 在 HTTP **GET** 或 **DELETE** 请求上, 将忽略 x-msg-persistence 头。
- 在 x-msg-require-headers 中指定, 将 HTTP 响应消息中的 x-msg-persistence 设置为消息的持久性。

priority:HTTP x-msg-priority entity-header

设置或返回消息优先级。

| 类型 | 描述 |
|----------------|---|
| HTTP 头名称 | x-msg-priority |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , x-msg-require-headers |
| 允许的值 | LOW 例如, <pre>x-msg-priority: LOW</pre> MEDIUM 此优先级等于 WebSphere MQ 优先级级别 4。例如, <pre>x-msg-priority: MEDIUM</pre> HIGH 例如, <pre>x-msg-priority: HIGH</pre> Integer value 范围在 0 到 9 之间的整数的字符串表示; 例如, <pre>x-msg-priority: 3</pre> AS_DESTINATION 仅适用于 POST 。 使用由消息提供程序确定的目标的缺省优先级。 注: 区分大小写 |
| 缺省值 | MEDIUM |

描述

- 在 HTTP **POST** 请求上设置时, 设置请求消息优先级。
- 在 HTTP **GET** 或 **DELETE** 请求上, 将忽略 x-msg-priority 头。
- 在 x-msg-require-headers 中指定, 将 HTTP 响应消息中的 x-msg-priority 设置为消息的优先级。

priority:HTTP x-msg-priority entity-header

设置或返回消息优先级。

| 类型 | 描述 |
|----------------|-------------------------------------|
| HTTP 头名称 | x-msg-priority |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , x-msg-require-headers |
| 允许的值 | LOW 例如, |

| 类型 | 描述 |
|-----|---|
| | <pre>x-msg-priority: LOW</pre> <p>MEDIUM 此优先级等于 WebSphere MQ 优先级级别 4。例如,</p> <pre>x-msg-priority: MEDIUM</pre> <p>HIGH 例如,</p> <pre>x-msg-priority: HIGH</pre> <p>Integer value 范围在 0 到 9 之间的整数的字符串表示; 例如,</p> <pre>x-msg-priority: 3</pre> <p>AS_DESTINATION 仅适用于 POST。 使用由消息提供程序确定的目标的缺省优先级。</p> <p>注: 区分大小写</p> |
| 缺省值 | MEDIUM |

描述

- 在 HTTP **POST** 请求上设置时, 设置请求消息优先级。
- 在 HTTP **GET** 或 **DELETE** 请求上, 将忽略 x-msg-priority 头。
- 在 x-msg-require-headers 中指定, 将 HTTP 响应消息中的 x-msg-priority 设置为消息的优先级。

replyTo:HTTP x-msg-replyTo entity-header

设置或返回消息应答队列和队列管理器名称。

| 类型 | 描述 |
|----------------|---|
| HTTP 头名称 | x-msg-replyTo |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , x-msg-require-headers |
| 允许的值 | <p>URI 点对点 URI; 例如,</p> <pre>x-msg-replyTo: /msg/queue/myReplyQueue x-msg-replyTo: /msg/queue/myReplyQueue@myReplyQueueManager</pre> <p>注: 区分大小写</p> |
| 缺省值 | MEDIUM |

描述

- 在 HTTP **POST** 请求上设置时, 请设置请求消息 replyTo 目标。
- 在 HTTP **GET** 或 **DELETE** 请求上, 将忽略 x-msg-replyTo 头。

- 在 `x-msg-require-headers` 中指定，将 HTTP 响应消息中的 `x-msg-replyTo` 设置为 消息的应答队列和队列管理器名称。

注: HTTP 响应中的 URI 可以包含 WebSphere MQ Bridge for HTTP 所连接到的队列管理器的名称。

服务器:HTTP 响应头

返回有关客户机所连接到的服务器和协议的信息。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | SERVER |
| HTTP 头类型 | 响应头 |
| 在 HTTP 请求消息中有效 | <code>x-msg-require-headers</code> |
| 返回的值 | <pre>WMQ-HTTP/1.1 JEE-Bridge/1.1</pre> <p>或</p> <pre>Server: Product-token WMQ-HTTP/1.1 JEE-Bridge/1.1</pre> |

描述

- 如果将 WebSphere MQ Bridge for HTTP 部署到应用程序服务器，那么会将 WebSphere MQ Bridge for HTTP 详细信息追加到服务器响应头。例如，部署到 WebSphere Application Server Community Edition 的 WebSphere MQ 网桥 (称为 Apache-Coyote) 提供了响应:

```
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
```

require-headers:HTTP x-msg-require-headers request-header

设置要在 HTTP 响应消息中返回的头。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | <code>x-msg-require-headers</code> |
| HTTP 头类型 | 请求头 |
| 在 HTTP 请求消息中有效 | POST, GET, DELETE |
| 允许的值 | <p>以逗号分隔的实体头名称列表:</p> <p>ALL</p> <p>ALL-USR</p> <p>class</p> <p>content-location</p> <p>correlId</p> <p>encoding</p> <p>expiry</p> <p>format</p> <p>msgId</p> <p>NO_require-headers</p> <p>persistence</p> <p>priority</p> |

| 类型 | 描述 |
|-----|---|
| | <p>replyTo server timestamp usr-property name</p> <p>例如,</p> <pre>x-msg-require-headers: msgId</pre> <p>或者,</p> <pre>x-msg-require-headers: expiry,correlId,timestamp</pre> <p>要请求特定属性:</p> <pre>x-msg-require-headers: usr-myCustomProperty</pre> <p>要请求所有属性:</p> <pre>x-msg-require-headers: ALL-USR, ALL</pre> |
| 缺省值 | NO_require-headers |

描述

- `x-msg-require-headers` 的值不区分大小写，但 ALL，NO_require-headers 和 ALL-USR 常量以及 `property-name` 变量的情况除外。

timestamp:HTTP x-msg-timestamp entity-header

返回消息时间戳记。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | x-msg-timestamp |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | x-msg-require-headers |
| 返回的值 | <p>HTTP-date 格式为; day , date month year time time-zone; 例如,</p> <pre>Sun, 06 Nov 1994 08:49:37 GMT</pre> <p>由 RFC 822 定义，并在 RFC 1123 中更新。</p> |
| 缺省值 | 不适用 |

描述

- 在 HTTP **POST**，**GET** 或 **DELETE** 请求上，将忽略 x-msg-timestamp 头。
- 在 x-msg-require-headers 中指定，将 HTTP 响应消息中的 x-msg-timestamp 设置为消息的时间戳记。

usr:HTTP x-msg-usr entity-header

设置或返回用户属性。

| 类型 | 描述 |
|----------------|---|
| HTTP 头名称 | x-msg-usr |
| HTTP 头类型 | 实体标题 |
| 在 HTTP 请求消息中有效 | POST , x-msg-require-headers |
| 允许的值 | 请参阅 第 1109 页的『语法』; 例如, <pre>x-msg-usr: myProp1;5;i1, x-msg-usr: myProp2;"My String";string</pre> |
| 缺省值 | 不适用 |

描述

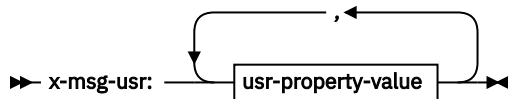
- 在 HTTP **POST** 请求上设置时, 设置请求消息用户属性。
- 在 HTTP **GET** 或 **DELETE** 请求上, 将忽略 x-msg-usr 头。
- 在 x-msg-require-headers 中指定, 将 HTTP 响应消息中的 x-msg-usr 设置为消息的用户属性。
- 可以在消息上设置多个属性。在单个 x-msg-usr 头中指定多个以逗号分隔的属性, 或者使用 x-msg-usr 头的两个或多个单独实例来指定这些属性。
- 您可以请求在对 **GET** 或 **DELETE** 请求的响应中返回特定属性。使用前缀 **usr-** 在请求的 x-msg-require-headers 头中指定属性的名称。例如,

```
x-msg-require-headers: usr-myProp1
```

- 要请求在响应中返回所有用户属性, 请使用 ALL-USR 常量。例如,

```
x-msg-require-headers: ALL-USR
```

语法



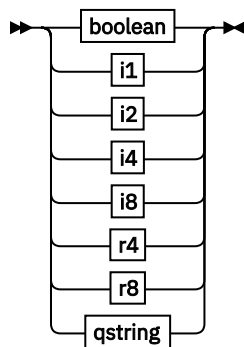
usr-property-value



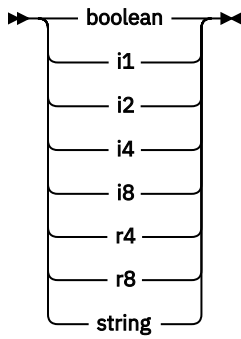
property-name



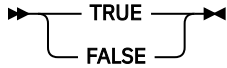
usr-value



usr-type



boolean



i1

▶ -128 — ≤n≤ — +127 ▶

i2

▶ -32768 — ≤n≤ — +32767 ▶

i4

▶ -2147483648 — ≤n≤ — +2147483647 ▶

i8

▶ -9223372036854775808 — ≤n≤ — +92233720368547750807 ▶

r4

▶ -1.4E-45 — ≤n≤ — +3.4028235E38 ▶

r8

▶ -4.9E-324 — ≤n≤ — +1.7976931348623157E308 ▶

qstring

▶ " — 字符串 — " ▶

wait:HTTP x-msg-wait request-header

设置在返回 HTTP 504 Gateway Timeout 响应消息之前等待消息到达的时间段。

| 类型 | 描述 |
|----------------|--|
| HTTP 头名称 | x-msg-wait |
| HTTP 头类型 | 请求头 |
| 在 HTTP 请求消息中有效 | GET, DELETE |
| 允许的值 | <p>NO_WAIT 例如,</p> <pre>x-msg-wait: NO_WAIT</pre> <p>Integer value WebSphere MQ Bridge for HTTP 等待消息到达的毫秒数; 例如,</p> <pre>x-msg-wait: 8</pre> |

| 类型 | 描述 |
|-----|---------|
| 缺省值 | NO_WAIT |

描述

- 在 HTTP **POST** 请求中，将忽略 `x-msg-wait` 头。
- 在 HTTP **GET** 或 **DELETE** 请求上，`x-msg-wait` 指定在返回 HTTP 504 Gateway Timeout 响应之前等待消息到达的时间。
- NO_WAIT 区分大小写。
- 缺省最大等待时间为 35000。您可以通过设置 servlet 的 `maximum_wait_time` 参数来更改缺省值。有关更多信息，请参阅 [安装，配置和验证 WebSphere MQ Bridge for HTTP](#) 部分。
- 如果设置的值大于 `maximum_wait_time`，那么将改为使用 `maximum_wait_time`。

HTTP 返回码

来自 WebSphere MQ Bridge for HTTP 的返回码列表

WebSphere MQ Bridge for HTTP 返回四种类型的错误:

Servlet 错误

MQHTTP0001 和 MQHTTP0002 是 Servlet 错误。它们已记录，但未返回到 HTTP 客户机。

成功的操作

200-299 范围内的 HTTP 状态码指示操作成功。

客户机错误数

400-499 范围内的 HTTP 状态码指示客户机错误。WebSphere MQ Bridge for HTTP 返回码 (范围为 MQHTTP40001 - MQHTTP49999) 对应于客户机错误。

服务器错误数

500-599 范围内的 HTTP 状态码指示客户机错误。WebSphere MQ Bridge for HTTP 返回码 (范围为 MQHTTP50001 - MQHTTP59999) 对应于服务器错误。

如果发生服务器错误，那么会将完整的堆栈跟踪输出到应用程序服务器错误日志。还会将堆栈跟踪返回到 HTTP 响应中的 HTTP 客户机。处理客户机应用程序中的堆栈跟踪，或将其提交给应用程序服务器管理员以解决问题。

如果堆栈跟踪包含资源适配器错误，请参阅资源适配器的文档。

返回码的字母列表

HTTP 200: OK

此类状态码指示已成功接收，理解和接受请求。

HTTP 状态码

200 OK

HTTP 204: 无内容

在请求中发送成功的 HTTP **GET** 或 **DELETE** 和 `x-msg-range: 0` 后发送。

HTTP 状态码

204 No Content

MQHTTP0001: 在 Servlet 上下文中未指定连接工厂

Servlet 错误

说明

Servlet 错误

HTTP 状态码

None

程序员响应

记录这些错误的位置特定于应用程序服务器。请参阅应用程序服务器的文档。

MQHTTP0002: 无法使用 JNDI 名称 *jndiNameTried* 获取 *queueOrTopic* 的连接管理器

Servlet 错误

说明

Servlet 错误

HTTP 状态码

None

程序员响应

记录这些错误的位置特定于应用程序服务器。请参阅应用程序服务器的文档。

MQHTTP40001: 保留

已保留

HTTP 状态码

400 Bad Request

MQHTTP40002: URI 对于 HTTP 的 WebSphere MQ 传输无效

HTTP 请求中指定的 URI 无效。

说明

HTTP 请求中指定的 URI 无效。

HTTP 状态码

400 Bad Request

程序员响应

请确认指定 URI 的格式和语法是否正确。

MQHTTP40003: URI 无效。@qmgr 仅在 POST 上有效

在不是 **POST** 请求的 HTTP 请求的 URI 中指定了 @qmgr URI 选项。

说明

在不是 **POST** 请求的 HTTP 请求的 URI 中指定了 @qmgr URI 选项。

HTTP 状态码

400 Bad Request

程序员响应

如果您尝试使用 **POST** 动词来放置消息，请将 HTTP 请求更改为 **POST** 请求。如果尝试使用 **DELETE** 或 **GET** 动词获取消息，请从 URI 中除去 @qmgr。

MQHTTP40004: 指定的 Content-Type 无效

在 **POST** 请求上指定的 Content-Type 头字段与 x-msg-class 头值不兼容。

说明

在 **POST** 请求上指定的 Content-Type 头字段与 x-msg-class 头值不兼容。

HTTP 状态码

400 Bad Request

程序员响应

将 Content-Type 头字段更改为受支持的字段。Content-Type 头必须与指定的 x-msg-class 头字段兼容。

MQHTTP40005: 错误消息头值

已使用对指定请求无效的值指定了受支持的头字段。

说明

已使用对指定请求无效的值指定了受支持的头字段。

HTTP 状态码

400 Bad Request

程序员响应

将为给定头字段指定的值更改为有效的值。检查指定值的大小写，因为某些头字段具有区分大小写的值。

MQHTTP40006: *Header_name* 不是有效的请求头

在 HTTP 请求消息中指定了仅在 HTTP 响应消息中有效的头。

说明

在 HTTP 请求消息中指定了仅在 HTTP 响应消息中有效的头。

HTTP 状态码

400 Bad Request

程序员响应

从 HTTP 请求中除去仅在 HTTP 响应中有效的任何头；例如， x-msg-timestamp。

MQHTTP40007: *Header_name* 仅在 ... 上有效

已在 HTTP 请求中指定了头，但头字段对于给定的请求动词无效。

说明

已在 HTTP 请求中指定了头，但头字段对于给定的请求动词无效。

HTTP 状态码

400 Bad Request

程序员响应

从 HTTP 请求中除去对给定请求动词无效的任何头。例如， x-msg-encoding 对 HTTP **POST** 请求有效，但对 HTTP **GET** 或 HTTP **DELETE** 请求无效。

MQHTTP40008: Header_name 最大长度为 ...

已超过给定头字段的最大长度。

说明

已超过给定头字段的最大长度。

HTTP 状态码

400 Bad Request

程序员响应

将头字段的值更改为在头字段允许的范围内的值。

MQHTTP40009: 头字段 header_field 对于 ... 无效

HTTP 请求中指定的头字段不受 WebSphere MQ Bridge for HTTP 所连接的消息传递提供程序支持。

说明

HTTP 请求中指定的头字段不受 WebSphere MQ Bridge for HTTP 所连接的消息传递提供程序支持。如果使用的消息传递提供程序无法支持 WebSphere MQ Bridge for HTTP 的所有功能，那么会发生此错误。

HTTP 状态码

400 Bad Request

程序员响应

从 HTTP 请求中除去不受支持的头。

MQHTTP40010: 无法解析具有 Content-Type content_type 的消息

HTTP 请求的内容与请求的 Content-Type 不兼容。

说明

HTTP 请求的内容与请求的 Content-Type 不兼容。常见原因的格式为 application/x-www-form-urlencoded 或 application/xml data。

HTTP 状态码

400 Bad Request

程序员响应

请更正 HTTP 请求的内容， 以使其采用请求的 Content-Type 的正确格式。

MQHTTP40301: 禁止您访问 ...

WebSphere MQ Bridge for HTTP 无法为指定的目标认证自身。

说明

WebSphere MQ Bridge for HTTP 无法为指定的目标认证自身。

HTTP 状态码

403 Forbidden

程序员响应

更改目标的认证属性，以便授权 WebSphere MQ Bridge for HTTP 连接到该目标。或者，指定 WebSphere MQ Bridge for HTTP 有权连接到的目标。

MQHTTP40302: 禁止您 ...

用于 HTTP 的 WebSphere MQ 网桥无法连接到队列管理器。

说明

用于 HTTP 的 WebSphere MQ 网桥无法连接到队列管理器。用于 HTTP 安全性配置的 WebSphere MQ 网桥不正确。

HTTP 状态码

403 Forbidden

程序员响应

更改队列管理器的认证配置，以便授权 WebSphere MQ Bridge for HTTP 与其连接。或者，配置 WebSphere MQ Bridge for HTTP 以连接到它有权连接到的队列管理器。

MQHTTP40401: 找不到目标 *destination_name*

HTTP 请求 URI 中指定的目标无法由 HTTP 的 WebSphere MQ 网桥找到。

说明

HTTP 请求 URI 中指定的目标无法由 HTTP 的 WebSphere MQ 网桥找到。

HTTP 状态码

404 Not found

程序员响应

请检查 HTTP 请求 URI 中指定的目标是否存在，或者指定备用目标。

MQHTTP40501: 方法 *method_name*

HTTP 请求中指定的方法不受 WebSphere MQ Bridge for HTTP 支持。

说明

HTTP 请求中指定的方法不受 WebSphere MQ Bridge for HTTP 支持。

HTTP 状态码

405 Method not allowed

程序员响应

将 HTTP 请求中指定的方法更改为 WebSphere MQ Bridge for HTTP 支持的方法。

MQHTTP41301: 要发布的消息对于目标而言过大

HTTP POST 请求 URI 中指定的目标无法接受与 HTTP 请求中指定的消息一样长的消息。

说明

HTTP POST 请求 URI 中指定的目标无法接受与 HTTP 请求中指定的消息一样长的消息。

HTTP 状态码

413 Request entity too large

程序员响应

减小 HTTP 请求中指定的消息的大小。或者，指定可支持所需长度的消息的目标。

MQHTTP41501: 介质类型字符集不受支持

WebSphere MQ Bridge for HTTP 不支持 Content-Type 头字段中指定的字符集。

说明

WebSphere MQ Bridge for HTTP 不支持 Content-Type 头字段中指定的字符集。

HTTP 状态码

415 Unsupported media type

程序员响应

将 Content-Type 头字段的字符集更改为 WebSphere MQ Bridge for HTTP 支持的字符集。

MQHTTP41502: Media-type *media-type* 不受支持 ...

对于指定的 HTTP 动词， WebSphere MQ Bridge for HTTP 不支持 HTTP 请求中指定的介质类型。

说明

对于指定的 HTTP 动词， WebSphere MQ Bridge for HTTP 不支持 HTTP 请求中指定的介质类型。

HTTP 状态码

415 Unsupported media type

程序员响应

将 HTTP 请求中指定的介质类型更改为指定 HTTP 动词的 WebSphere MQ Bridge for HTTP 支持的介质类型。

MQHTTP41503: Media-type *media-type* 不受支持 ...

对于指定的 x-msg-class 头字段， HTTP 请求中指定的介质类型不受 WebSphere MQ bridge for HTTP 支持。

说明

对于指定的 x-msg-class 头字段， HTTP 请求中指定的介质类型不受 WebSphere MQ bridge for HTTP 支持。

HTTP 状态码

415 Unsupported media type

程序员响应

将 HTTP 请求中指定的介质类型更改为 WebSphere MQ Bridge for HTTP 针对指定的 x-msg-class 头字段支持的介质类型。

MQHTTP41701: 不支持 HTTP 头 Expect

WebSphere MQ Bridge for HTTP 不支持 Expect 头字段。

说明

已在 HTTP 请求中指定 Expect 头。 WebSphere MQ Bridge for HTTP 不支持 Expect 头字段。

HTTP 状态码

417 Expectation failed

程序员响应

从 HTTP 请求中除去 Expect 头。

MQHTTP50001: 发生了意外问题 ...

WebSphere MQ Bridge for HTTP 中发生错误。

说明

WebSphere MQ Bridge for HTTP 中发生错误。

HTTP 状态码

500 Internal server error

程序员响应

请与 WebSphere MQ Bridge for HTTP 的系统管理员联系。

MQHTTP50201: WebSphere MQ Bridge for HTTP 与队列管理器之间发生错误

用于 HTTP 的 WebSphere MQ 网桥与队列管理器之间发生错误

说明

用于 HTTP 的 WebSphere MQ 网桥与队列管理器之间发生错误

HTTP 状态码

502 Bad Gateway

程序员响应

请与 WebSphere MQ Bridge for HTTP 的系统管理员联系。

MQHTTP50401: 消息检索超时

在超时时间段内，未返回与 HTTP **GET** 或 HTTP **DELETE** 中的指定请求参数匹配的消息。

说明

在超时时间段内，未返回与 HTTP **GET** 或 HTTP **DELETE** 中的指定请求参数匹配的消息。返回码指示在 HTTP 请求的生命周期内，任何时候都没有合适的消息可用。

HTTP 状态码

504 Gateway timeout

程序员响应

如果需要消息，请检查 HTTP 请求的头字段，例如 `x-msg-correlId` 和 `x-msg-msgid`。请检查 HTTP 请求 URI 中指定的目标是否正确。尝试使用 `x-msg-wait` 头字段来延长 HTTP 请求的等待时间。

MQHTTP50501: HTTP 1.1 及更高版本 ...

WebSphere MQ Bridge for HTTP 不支持 HTTP 请求中使用的 HTTP 协议。

说明

WebSphere MQ Bridge for HTTP 不支持 HTTP 请求中使用的 HTTP 协议。

HTTP 状态码

505 HTTP version not supported

程序员响应

更改 HTTP 请求以使用 HTTP 协议 V1.1 或更高版本。

WebSphere Bridge for HTTP 的消息类型和消息映射

WebSphere MQ Bridge for HTTP 支持四个消息类 TEXT, BYTES, STREAM 和 MAP。消息类映射到 JMS 消息类型和 HTTP Content-Type。

HTTP POST

到达目标的消息类型取决于 `x-msg-class` 头的值或 HTTP 请求的 Content-Type。第 1118 页的表 604 显示对应于每个 `x-msg-class` 的 HTTP Content-Type 类型。可以使用任一字段来设置消息类型和消息格式。如果设置了这两个字段，并且设置不一致，那么将返回 Bad Request exception (HTTP 400, MQHTTP200004)。

| <code>x-msg-class</code> | HTTP 内容类型 |
|--------------------------|---|
| BYTES | application/octet-stream application/xml |
| TEXT | text/* |
| MAP | application/x-www-form-urlencoded application/xml (可选) |
| STREAM | application/xml (可选) |

如果在 MQRFH2 头中设置了 JMS 消息类型，那么将根据第 1118 页的表 605 对其进行映射。

| <code>x-msg-class</code> | JMS 消息类型 |
|--------------------------|------------|
| BYTES | jms_bytes |
| TEXT | jms_text |
| MAP | jms_map |
| STREAM | jms_stream |

始终为消息类 MAP 或 STREAM 设置 JMS 消息类型。并非总是为消息类 BYTES 或 TEXT 设置此值。如果要为请求创建 MQRFH2，那么将始终设置 JMS 消息类型。否则，如果未创建任何 MQRFH2，那么不会设置 JMS 消息类型。如果在请求中使用 `x-msg-usr` 头设置了用户属性，那么将创建 MQRFH2。

如果设置了 JMS 消息类型，那么消息格式设置为 MQFMT_NONE，请参阅第 1119 页的表 607:

表 606: *x-msg-class* 与 WebSphere MQ 消息格式之间的映射

| x-msg-class | 消息中存在 MQRFH2 的消息格式 | 消息中不存在 <i>no MQRFH2</i> 的消息格式 |
|--------------------|--------------------|-------------------------------|
| BYTES | MQFMT_NONE | MQFMT_NONE |
| TEXT | MQFMT_NONE | MQFMT_STRING |
| MAP | MQFMT_NONE | 不可用 |
| STREAM | MQFMT_NONE | 不可用 |

HTTP GET 或 DELETE

检索的消息类型或格式确定 HTTP 响应的 *x-msg-class* 头和 Content-Type 的值。仅当在 *x-msg-headers* 请求中请求时，才会返回 *x-msg-class* 头。

第 1119 页的表 607 描述了 *x-msg-class* 和 Content-Type 之间的映射，以及从队列或主题检索的消息类型。

表 607: 将消息类型映射到 *x-msg-class* 和 Content-Type

| 消息格式 | JMS 消息类型 | x-msg-class | 内容类型 |
|------------------------|------------|--------------------|--------------------------|
| 除 MQFMT_STRING 以外的任何内容 | None | BYTES | application/octet-stream |
| MQFMT_STRING | None | 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 |

MAP 和 STREAM 消息类序列化

在 HTTP 响应中将 MAP 和 STREAM 消息类序列化回客户机的方式与将消息序列化到队列的方式相同。

对于 MAP，XML 名称，类型和值三元组将编码为：

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

STREAM 类似于 MAP，但它没有元素名称：

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

注: datatype 是为定义用户定义的属性而定义的其中一种数据类型，在第 1108 页的『usr:HTTP x-msg-usr entity-header』中列出。对于字符串元素，将省略属性 dt="string"，因为缺省数据类型为 string。

URI 格式

WebSphere MQ Bridge for HTTP 拦截的 URI。


```
public static int Reason {get;}
```

来自第一个错误或警告的原因码。

```
public static int PutSuccessCount {get;}
```

成功的异步 MQI 放置调用数。

```
public static int PutWarningCount {get;}
```

成功且带有警告的异步 MQI put 调用数。

```
public static int PutFailureCount {get;}
```

失败的异步 MQI 放置调用数。

```
public static int ObjectType {get;}
```

第一个错误的对象类型。可能的值如下所示：

- MQC.MQOT_ALIAS_Q
- MQC.MQOT_LOCAL_Q
- MQC.MQOT_MODEL_Q
- MQC.MQOT_Q
- MQC.MQOT_REMOTE_Q
- MQC.MQOT_TOPIC
- 0, 表示不返回任何对象

```
public static string ObjectName {get;}
```

对象名称。

```
public static string ObjectQMgrName {get;}
```

对象队列管理器名称。

```
public static string ResolvedObjectName {get;}
```

已解析的对象名。

```
public static string ResolvedObjectQMgrName {get;}
```

已解析的对象队列管理器名称。

构造函数

```
public MQAsyncStatus() throws MQException;
```

构造方法，根据需要构造具有初始化为零或空白的字段的对象。

MQAuthenticationInformationRecord.NET 类

使用 MQAuthenticationInformationRecord 来指定有关要在 WebSphere MQ SSL 客户机连接中使用的认证程序的信息。MQAuthenticationInformationRecord 封装了认证信息记录 MQAIR。

类

```
System.Object  
└─ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```

- [第 1122 页的『属性』](#)
- [第 1122 页的『构造函数』](#)

属性

获取属性时抛出 `MQException` 的测试。

```
public long Version {get; set;}
```

结构版本号。

```
public long AuthInfoType {get; set;}
```

认证信息的类型。此属性必须设置为下列其中一个值：

- `OCSP` -使用 `OCSP` 完成证书撤销状态检查。
- `CRLLDAP` -使用 `LDAP` 服务器上的证书撤销列表完成证书撤销状态检查。

```
public string AuthInfoConnName {get; set;}
```

运行 `LDAP` 服务器的主机的 `DNS` 名称或 `IP` 地址，带有可选端口号。此关键字是必需的。

```
public string LDAPPASSWORD {get; set;}
```

与访问 `LDAP` 服务器的用户的专有名称关联的密码。仅当 `AuthInfoType` 设置为 `CRLLDAP` 时，此属性才适用。

```
public string LDAPUserName {get; set;}
```

访问 `LDAP` 服务器的用户的专有名称。设置此属性时，将自动正确设置 `LDAPUserNameLength` 和 `LDAPUserNamePtr`。仅当 `AuthInfoType` 设置为 `CRLLDAP` 时，此属性才适用。

```
public string OCSPResponderURL {get; set;}
```

可联系 `OCSP` 响应程序的 `URL`。仅当 `AuthInfoType` 设置为 `OCSP` 时，此属性才适用

此字段区分大小写。它必须以小写的字符串 `http://` 开头。其余 `URL` 可能区分大小写，具体取决于 `OCSP` 服务器实现。

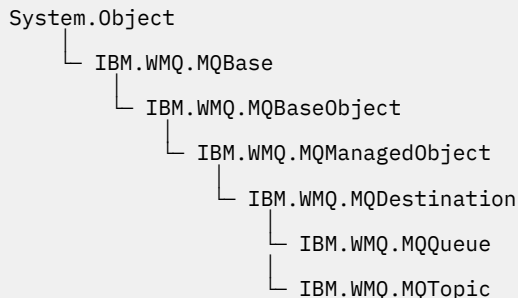
构造函数

```
MQAuthenticationInformationRecord();
```

MQDestination.NET 类

使用 `MQDestination` 来访问 `MQQueue` 和 `MQTopic` 通用的方法。`MQDestination` 是抽象基类，无法实例化。

类



```
public class IBM.WMQ.MQDestination extends IBM.WMQ.MQManagedObject;
```

- [第 1123 页的『属性』](#)
- [第 1123 页的『方法』](#)
- [第 1124 页的『构造函数』](#)

属性

获取属性时抛出 `MQException` 的测试。

```
public DateTime CreationDateTime {get;}
```

创建队列或主题的时间和日期。此属性最初包含在 `MQQueue` 中，但已移动到基本 `MQDestination` 类中。

无缺省值。

```
public int DestinationType {get;}
```

用于描述正在使用的目标类型的整数值。从子类构造函数 `MQQueue` 或 `MQTopic` 初始化，此值可以采用下列其中一个值：

- `MQOT_Q`
- `MQOT_TOPIC`

无缺省值。

方法

```
public void Get(MQMessage message);
```

```
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);
```

```
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int MaxMsgSize);
```

抛出 `MQException`。

如果目标是 `MQQueue` 对象，那么从队列中获取消息；如果目标是 `MQTopic` 对象，那么从主题中获取消息，并使用 `MQGetMessageOptions` 的缺省实例来执行获取。

如果 `get` 失败，那么 `MQMessage` 对象保持不变。如果成功，那么 `MQMessage` 的消息描述符和消息数据部分将替换为来自入局消息的消息描述符和消息数据。

从特定 `MQQueueManager` 对 WebSphere MQ 的所有调用都是同步的。因此，如果执行获取并等待，那么将阻止使用相同 `MQQueueManager` 的所有其他线程进行进一步的 WebSphere MQ 调用，直到完成 `Get` 调用为止。如果需要多个线程来同时访问 WebSphere MQ，那么每个线程都必须创建自己的 `MQQueueManager` 对象。

message

包含消息描述符和返回的消息数据。消息描述符中的某些字段是输入参数。确保根据需要设置 `MessageId` 和 `CorrelationId` 输入参数非常重要。

对于在 `MQGM_SYNCPOINT` 下接收到的消息，可重新连接的客户机在成功重新连接后返回原因码 `MQRC_BACKED_OUT`。

getMessageOptions

控制获取操作的选项。

使用选项 `MQC.MQGMO_CONVERT` 可能会导致在从单字节字符代码转换为双字节代码时发生异常，原因码为 `MQC.MQRC_CONVERTED_STRING_TOO_BIG`。在这种情况下，会将消息复制到缓冲区中而不进行转换。

如果未指定 `getMessageOptions`，那么使用的消息选项为 `MQGMO_NOWAIT`。

如果在可重新连接的客户机中使用 `MQGMO_LOGICAL_ORDER` 选项，那么将返回 `MQRC_RECONNECT_INCOMPATIBLE` 原因码。

MaxMsgSize

此消息对象要接收的最大消息。如果队列上的消息大于此大小，那么将发生以下两种情况之一：

- 如果在 `MQGetMessageOptions` 对象中设置了 `MQGMO_ACCEPT_TRUNCATED_MSG` 标志，那么将使用尽可能多的消息数据来填充消息。抛出异常时带有 `MQCC_WARNING` 完成代码和 `MQRC_TRUNCATED_MSG_ACCEPTED` 原因码。

- 如果未设置 MQGMO_ACCEPT_TRUNCATED_MSG 标志，那么消息将保留在队列中。抛出异常时带有 MQCC_WARNING 完成代码和 MQRC_TRUNCATED_MSG_FAILED 原因码。

如果未指定 *MaxMsgSize*，那么将检索整个消息。

```
public void Put(MQMessage message);  
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

抛出 MQException。

如果目标是 MQQueue 对象，那么将消息放入队列；如果目标是 MQTopic 对象，那么将消息发布到主题。

完成 Put 调用后对 MQMessage 对象的修改不会影响 WebSphere MQ 队列或发布主题上的实际消息。

Put 会更新 MQMessage 对象的 MessageId 和 CorrelationId 属性，并且不会清除消息数据。进一步的 Put 或 Get 调用引用 MQMessage 对象中的更新信息。例如，在以下代码片段中，第一条消息包含 a 和第二条 ab。

```
msg.WriteString("a");  
q.Put(msg,pmo);  
msg.WriteString("b");  
q.Put(msg,pmo);
```

message

包含消息描述符数据和要发送的消息的 MQMessage 对象。由于此方法，可以更改消息描述符。在此方法完成后，消息描述符中的值是已放入队列或已发布到主题的值。

以下原因码将返回到可重新连接的客户机：

- MQRC_CALL_INTERRUPTED：如果在对持久消息运行 Put 调用时连接中断，并且重新连接成功。
- MQRC_NONE (如果在对非持久消息运行 Put 调用时连接成功) (请参阅 [应用程序恢复](#))。

putMessageOptions

用于控制 put 操作的选项。

如果未指定 *putMessageOptions*，那么将使用 MQPutMessageOptions 的缺省实例。

如果在可重新连接的客户机中使用 MQPMO_LOGICAL_ORDER 选项，那么将返回 MQRC_RECONNECT_INCOMPATIBLE 原因码。

注：为了实现简单性和性能，如果要将单个消息放入队列，请使用 MQQueueManager.Put 对象。您应该具有此对象的 MQQueue 对象。

构造函数

MQDestination 是抽象基类，无法实例化。使用 MQQueue 和 MQTopic 构造函数或使用 MQQueueManager.AccessQueue 和 MQQueueManager.AccessTopic methods 访问目标。

MQEnvironment.NET 类

使用 MQEnvironment 来控制如何调用 MQQueueManager 构造函数以及选择 WebSphere MQ MQI 客户机连接。MQEnvironment 类包含用于控制 WebSphere MQ 行为的属性。

类

```
System.Object  
└─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```

- [第 1125 页的『属性-仅限客户机』](#)
- [第 1125 页的『属性』](#)
- [第 1126 页的『构造函数』](#)

属性-仅限客户机

获取属性时抛出 MQException 的测试。

public static int CertificateValPolicy {get; set;}

设置使用哪个 SSL/TLS 证书验证策略来验证从远程伙伴系统接收的数字证书。有效值包括：

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

public static ArrayList EncryptionPolicySuiteB {get; set;}

设置符合 Suite B 的密码术级别。有效值包括：

- MQC.MQ_SUITE_B_NONE -这是缺省值。
- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

public static string Channel {get; set;}

用于连接到目标队列管理器的通道的名称。在以客户机方式实例化 MQQueueManager 实例之前，必须设置通道属性。

public static int FipsRequired {get; set;}

如果在 WebSphere MQ 中执行密码术，请指定 MQC.MQSSL_FIPS_YES 以仅使用 FIPS 认证的算法。缺省值为 MQC.MQSSL_FIPS_NO。

如果配置了加密硬件，那么使用的加密模块是由硬件产品提供的模块。根据正在使用的硬件，这些硬件可能未通过 FIPS 认证到特定级别。

public static string Hostname {get; set;}

WebSphere MQ 服务器所在计算机的 TCP/IP 主机名。如果未设置主机名，并且未设置覆盖属性，那么将使用服务器绑定方式来连接到本地队列管理器。

public static int Port {get; set;}

要连接到的端口。这是 WebSphere MQ 服务器侦听入局连接请求的端口。缺省值为 1414。

public static string SSLCipherSpec {get; set;}

将 SSLCipherSpec 设置为 SVRCONN 通道上设置的 CipherSpec 值，以对连接启用 SSL。缺省值为 Null，并且未对连接启用 SSL。

public static string sslPeerName {get; set;}

专有名称模式。如果设置了 sslCipherSpec，那么可以使用此变量来确保使用正确的队列管理器。如果设置为 null (缺省值)，那么不会执行队列管理器的 DN。如果 sslCipherSpec 为空，那么将忽略 sslPeerName。

属性

获取属性时抛出 MQException 的测试。

public static ArrayList HdrCompList {get; set;}

头数据压缩列表

public static int KeyResetCount {get; set;}

指示在重新协商密钥之前在 SSL 对话中发送和接收的未加密字节数。

public static ArrayList MQAIRArray {get; set;}

MQAuthenticationInformationRecord 对象的数组。

public static ArrayList MsgCompList {get; set;}

消息数据压缩列表

public static string Password {get; set;}

要认证的密码。将通过设置此 "密码" 属性来填充从 MQCSP 结构引用的密码。

public static string ReceiveExit {get; set;}

接收出口允许您检查和更改从队列管理器接收的数据。它通常与队列管理器上的相应发送出口配合使用。如果 ReceiveExit 设置为空，那么不会调用接收出口。

public static string ReceiveUserData {get; set;}

与接收出口关联的用户数据。限制为 32 个字符。

public static string SecurityExit {get; set;}

安全出口允许您定制尝试连接到队列管理器时发生的安全流。如果 SecurityExit 设置为空，那么不会调用安全出口。

public static string SecurityUserData {get; set;}

与安全出口关联的用户数据。限制为 32 个字符。

public static string SendExit {get; set;}

发送出口允许您检查或更改发送到队列管理器的数据。它通常与队列管理器上的相应接收出口配合使用。如果 SendExit 设置为空，那么不会调用发送出口。

public static string SendUserData {get; set;}

与发送出口关联的用户数据。限制为 32 个字符。

public static string SharingConversations {get; set;}

当来自 .NET 应用程序的连接未使用客户机通道定义表 (CCDT) 时，将使用 SharingConversations 字段。

SharingConversations 确定可以在与此连接关联的套接字上共享的最大对话数。

值 0 表示通道在对话共享，预读和脉动信号方面的操作与 WebSphere MQ 版本 7.0 之前的操作相同。

在将 WebSphere MQ 队列管理器实例化时，该字段将作为 SHARING_CONVERSATIONS_PROPERTY 传递到属性的散列表中。

如果未指定 SharingConversations，那么将使用缺省值 10。

public static string SSLCryptoHardware {get; set;}

设置配置系统上存在的加密硬件所需的参数字符串的名称。如果 sslCipherSpec 为空，那么将忽略 SSLCryptoHardware。

public static string SSLKeyRepository {get; set;}

设置密钥存储库的标准文件名。

如果 SSLKeyRepository 设置为 null (缺省值)，那么将使用证书 MQSSLKEYR 环境变量来查找密钥存储库。如果 sslCipherSpec 为空，那么将忽略 SSLCryptoHardware。

注：.kdb 扩展名是文件名的必需部分，但不包含在参数值中。您指定的目录必须存在。WebSphere MQ 在首次访问新的密钥存储库时创建该文件，除非该文件已存在。

public static string UserId {get; set;}

要认证的用户标识。将通过设置 UserId 来填充从 MQCSP 结构引用的用户标识。使用 API 或安全性出口认证 UserId。

构造函数

public MQEnvironment()

MQException.NET 类

使用 MQException 可查找失败的 WebSphere MQ 函数的完成代码和原因码。每当发生 WebSphere MQ 错误时，都会抛出 MQException。

类

```

System.Object
├── System.Exception
│   └── System.ApplicationException
│       └── IBM.WMQ.MQException

```

```
public class IBM.WMQ.MQException extends System.ApplicationException;
```

- [第 1127 页的『属性』](#)
- [第 1127 页的『构造函数』](#)

属性

```
public int CompletionCode {get; set;}
```

与错误关联的 WebSphere MQ 完成代码。可能的值为：

- MQException.MQCC_OK
- MQException.MQCC_WARNING
- MQException.MQCC_FAILED

```
public int ReasonCode {get; set;}
```

WebSphere MQ 原因码描述错误。

构造函数

```
public MQException(int completionCode, int reasonCode)
```

completionCode

WebSphere MQ 完成代码。

reasonCode

WebSphere MQ 完成代码。

MQGetMessageOptions.NET 类

使用 MQGetMessageOptions 来指定如何检索消息。它会修改 MQDestination.Get 的行为。

类

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQGetMessageOptions

```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```

- [第 1127 页的『属性』](#)
- [第 1130 页的『构造函数』](#)

属性

注: 此类中某些可用选项的行为取决于使用这些选项的环境。这些元素标有星号*。

获取属性时抛出 MQException 的测试。

```
public int GroupStatus {get;}*
```

GroupStatus 指示检索到的消息是否在组中，以及它是否是组中的最后一个消息。可能的值为：

MQC.MQGS_LAST_MSG_IN_GROUP

消息是组中的最后一条或唯一一条消息。

MQC.MQGS_MSG_IN_GROUP

消息位于组中，但不是组中的最后一个消息。

MQC.MQGS_NOT_IN_GROUP

消息不在组中。

public int MatchOptions {get; set;}*

MatchOptions 确定如何选择消息。可以设置以下匹配选项:

MQC.MQMO_MATCH_CORREL_ID

要匹配的相关标识。

MQC.MQMO_MATCH_GROUP_ID

要匹配的组标识。

MQC.MQMO_MATCH_MSG_ID

要匹配的消息标识。

MQC.MQMO_MATCH_MSG_SEQ_NUMBER

匹配消息序号。

MQC.MQMO_NONE

不需要匹配。

public int Options {get; set;}*

选项 控制 MQQueue.get 的操作。可以指定下列任何值。如果需要多个选项，那么可以使用按位 OR 运算符来添加或组合值。

MQC.MQGMO_ACCEPT_TRUNCATED_MSG

允许截断消息数据。

MQC.MQGMO_ALL_MSGS_AVAILABLE*

仅当组中的所有消息都可用时，才从组中检索消息。

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*

仅当组中的所有段都可用时，才检索逻辑消息的段。

MQC.MQGMO_BROWSE_FIRST

从队列开始浏览。

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*

浏览光标下的浏览消息。

MQC.MQGMO_BROWSE_NEXT

从队列中的当前位置进行浏览。

MQC.MQGMO_COMPLETE_MSG*

仅检索完整的逻辑消息。

MQC.MQGMO_CONVERT

请求要转换的应用程序数据，以符合 MQMessage 的 CharacterSet 和 Encoding 属性，然后再将数据复制到消息缓冲区中。由于从消息缓冲区检索数据时也会应用数据转换，因此应用程序不会设置此选项。

使用此选项可能会在从单字节字符集转换为双字节字符集时导致问题。而是在传递消息后使用 readString, readLine 和 writeString 方法执行转换。

MQC.MQGMO_FAIL_IF QUIESCING

如果队列管理器正在停顿，那么失败。

MQC.MQGMO_LOCK*

锁定浏览的消息。

MQC.MQGMO_LOGICAL_ORDER*

以逻辑顺序返回组中的消息以及逻辑消息段。

如果在可重新连接的客户机中使用 MQGMO_LOGICAL_ORDER 选项，那么会将 MQRC_RECONNECT_INCOMPATIBLE 原因码返回到应用程序。

MQC.MQGMO_MARK_SKIP_BACKOUT*

允许回退工作单元而不恢复队列上的消息。

MQC.MQGMO_MSG_UNDER_CURSOR

在浏览光标下获取消息。

MQC.MQGMO_NONE

未指定其他选项; 所有选项都采用其缺省值。

MQC.MQGMO_NO_PROPERTIES

不检索消息的属性, 消息描述符 (或扩展) 中包含的属性除外。

MQC.MQGMO_NO_SYNCPOINT

在没有同步点控制的情况下获取消息。

MQC.MQGMO_NO_WAIT

如果没有合适的消息, 请立即返回。

MQC.MQGMO_PROPERTIES_AS_Q_DEF

检索由 MQQueue 的 PropertyControl 属性定义的消息属性。对消息描述符或扩展中的消息属性的访问不受 PropertyControl 属性的影响。

MQC.MQGMO_PROPERTIES_COMPATIBILITY

在 MQRFH2 头中检索前缀为 mcd, jms, usr 或 mqext 的消息属性。将废弃消息的其他属性 (消息描述符或扩展中包含的属性除外)。

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2

检索 MQRFH2 头中的消息属性 (消息描述符或扩展中包含的属性除外)。在期望检索属性但无法更改为使用消息句柄的应用程序中使用 MQC.MQGMO_PROPERTIES_FORCE_MQRFH2。

MQC.MQGMO_PROPERTIES_IN_HANDLE

使用 MsgHandle 检索消息属性。

MQC.MQGMO_SYNCPOINT

在同步点控制下获取消息。该消息被标记为对其他应用程序不可用, 但仅当落实工作单元时才会从队列中删除该消息。如果回退工作单元, 那么该消息将再次可用。

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*

获取具有同步点控制的消息 (如果消息是持久消息)。

MQC.MQGMO_UNLOCK*

解锁先前锁定的消息。

MQC.MQGMO_WAIT

等待消息到达。

public string ResolvedQueueName {get;}

队列管理器将 ResolvedQueueName 设置为从中检索消息的队列的局部名。ResolvedQueue 名称与用于打开队列 (如果打开了别名队列或模型队列) 的名称不同。

public char Segmentation {get;}*

分段 指示您是否可以允许对检索到的消息进行分段。可能的值为:

MQC.MQSEG_INHIBITED

不允许分段。

MQC.MQSEG_ALLOWED

允许分段

public byte SegmentStatus {get;}*

SegmentStatus 是一个输出字段, 用于指示检索的消息是否为逻辑消息的段。如果消息是段, 那么标志指示它是否是最后一个段。可能的值为:

MQC.MQSS_LAST_SEGMENT

消息是逻辑消息的最后或唯一一段。

MQC.MQSS_NOT_A_SEGMENT

消息不是段。

MQC.MQSS_SEGMENT

消息是一个段, 但不是逻辑消息的最后一个段。

```
public int WaitInterval {get; set;}
```

WaitInterval 是 MQQueue.get 调用等待合适消息到达的最长时间 (以毫秒为单位)。将 WaitInterval 与 MQC.MQGMO_WAIT 配合使用。设置值 MQC.MQWI_UNLIMITED 以等待无限的消息时间。

构造函数

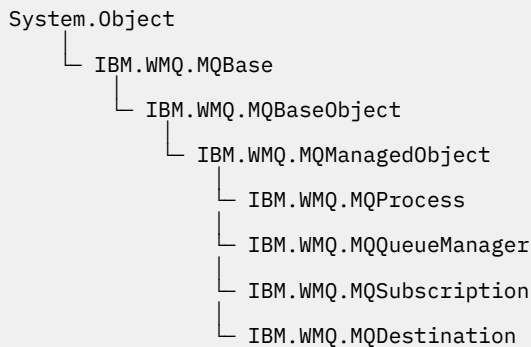
```
public MQGetMessageOptions()
```

构造新的 MQGetMessageOptions 对象, 其中选项设置为 MQC.MQGMO_NO_WAIT, WaitInterval 设置为零, ResolvedQueueName 设置为空白。

MQManagedObject .NET 类

使用 MQManagedObject 可查询和设置 MQDestination, MQProcess, MQQueueManager 和 MQSubscription 的属性。MQManagedObject 是这些类的超类。

类



```
public class IBM.WMQ.MQManagedObject extends IBM.WMQ.MQBaseObject;
```

- [第 1130 页的『属性』](#)
- [第 1131 页的『方法』](#)
- [第 1132 页的『构造函数』](#)

属性

获取属性时抛出 MQException 的测试。

```
public string AlternateUserId {get; set;}
```

打开资源时设置的备用用户标识 (如果有)。针对打开的对象发出 AlternateUserId.set 时会被忽略。AlternateUserId 对于预订无效。

```
public int CloseOptions {get; set;}
```

设置此属性以控制关闭资源的方式。缺省值为 MQC.MQCO_NONE。MQC.MQCO_NONE 是除永久动态队列, 临时动态队列, 预订以及创建这些资源的对象所访问的主题以外的所有资源的唯一允许值。

对于队列和主题, 允许使用以下附加值:

MQC.MQCO_DELETE

如果没有消息, 请删除队列。

MQC.MQCO_DELETE_PURGE

删除队列, 清除队列上的任何消息。

MQC.MQCO QUIESCE

请求关闭队列, 如果保留任何消息 (允许在最终关闭之前检索这些消息), 那么接收到警告。

对于预订, 允许使用以下附加值:

MQC.MQCO_KEEP_SUB

未删除预订。仅当原始预订是持久预订时，此选项才有效。MQC.MQCO_KEEP_SUB 是持久主题的缺省值。

MQC.MQCO_REMOVE_SUB

已删除预订。MQC.MQCO_REMOVE_SUB 是非持久非受管主题的缺省值。

MQC.MQCO_PURGE_SUB

已删除预订。MQC.MQCO_PURGE_SUB 是非持久受管主题的缺省值。

public MQQueueManager ConnectionReference {get;}

此资源所属的队列管理器。

public string MQDescription {get;}

队列管理器持有的资源的描述。MQDescription 返回预订和主题的空字符串。

public boolean IsOpen {get;}

指示资源当前是否处于打开状态。

public string Name {get;}

资源名称。该名称是访问方法上提供的名称，或者是队列管理器为动态队列分配的名称。

public int OpenOptions {get; set;}

OpenOptions 是在打开 WebSphere MQ 对象时设置的。将忽略 OpenOptions.set 方法，并且不会导致错误。预订没有 OpenOptions。

方法

public virtual void Close();

抛出 MQException。

关闭对象。在调用 Close 之后，不允许对此资源执行任何进一步的操作。要更改 Close 方法的行为，请设置 closeOptions 属性。

public string GetAttributeString(int selector, int length);

抛出 MQException。

获取属性字符串。

selector

指示正在查询哪个属性的整数。

length

整数，指示所需字符串的长度。

public void Inquire(int[] selectors, int[] intAttrs, byte[] charAttrs);

抛出 MQException。

返回整数数组和一组字符串，其中包含队列，进程或队列管理器的属性。要查询的属性在选择器数组中指定。

注：可以使用 MQManagedObject，MQQueue 和 MQQueueManager 中定义的 Get 方法来查询许多更常见的属性。

selectors

整数数组，用于标识具有要查询的值的属性。

intAttrs

返回整数属性值的数组。返回整数属性值的顺序与选择器数组中的整数属性选择器相同。

charAttrs

用于返回和并置字符属性的缓冲区。字符属性的返回顺序与选择器数组中的字符属性选择器的返回顺序相同。每个属性字符串的长度都是固定的。

public void Set(int[] selectors, int[] intAttrs, byte[] charAttrs);

抛出 MQException。

设置选择器的向量中定义的属性。要设置的属性在选择器数组中指定。

selectors

整数数组，用于标识具有要设置的值的属性。

intAttrs

要设置的整数属性值的数组。这些值必须与选择器数组中的整数属性选择器的顺序相同。

charAttrs

要设置的字符属性在其中并置的缓冲区。这些值必须与选择器数组中的字符属性选择器的顺序相同。每个字符属性的长度是固定的。

```
public void SetAttributeString(int selector, string value, int length);
```

抛出 MQException。

设置属性字符串。

selector

指示正在设置哪个属性的整数。

value

要设置为属性值的字符串。

length

整数，指示所需字符串的长度。

构造函数

```
protected MQManagedObject()
```

构造方法。此对象是无法单独实例化的抽象基类。

MQMessage .NET 类

使用 MQMessage 可访问 WebSphere MQ 消息的消息描述符和数据。MQMessage 封装了 WebSphere MQ 消息。

类

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQMessage
```

```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

创建 MQMessage 对象，然后使用 Read 和 Write 方法在消息与应用程序中的其他对象之间传输数据。使用 MQDestination, MQQueue 和 MQTopic 类的 Put 和 Get 方法发送和接收 MQMessage 对象。

使用 MQMessage 的属性获取和设置消息描述符的属性。使用 SetProperty 和 GetProperty 方法设置和获取扩展消息属性。

- [第 1132 页的『属性』](#)
- [第 1137 页的『Read 和 Write 消息方法』](#)
- [第 1140 页的『缓冲区方法』](#)
- [第 1141 页的『属性方法』](#)
- [第 1142 页的『构造函数』](#)

属性

获取属性时抛出 MQException 的测试。

```
public string AccountingToken {get; set;}
```

消息的部分身份上下文；它帮助应用程序对由于消息而完成的工作收取费用。缺省值为 MQC.MQACT_NONE。

public string ApplicationIdData {get; set;}

消息的部分身份上下文。ApplicationId 数据是由应用程序套件定义的信息，可用于提供有关消息或其发起方的其他信息。缺省值为 ""。

public string ApplicationOriginData {get; set;}

应用程序定义的信息，可用于提供有关消息来源的其他信息。缺省值为 ""。

public int BackoutCount {get;}

先前作为工作单元一部分由 MQQueue.Get 调用返回和回退消息的次数。缺省值为零。

public int CharacterSet {get; set;}

消息中字符数据的编码字符集标识。

设置 CharacterSet 以标识消息中字符数据的字符集。获取 CharacterSet 以了解使用了哪些字符集对消息中的字符数据进行编码。

.NET 应用程序始终以 Unicode 运行，而在其他环境中，应用程序以与队列管理器运行时使用的字符集相同的字符集运行。

ReadString 和 ReadLine 方法为您将消息中的字符数据转换为 Unicode。

WriteString 方法从 Unicode 转换为以 CharacterSet 编码的字符集。如果 CharacterSet 设置为其缺省值 MQC.MQCCSI_Q_MGR(即 0)，那么不会进行任何转换，并且 CharacterSet 设置为 1200。如果将 CharacterSet 设置为其他某个值，那么 WriteString 会从 Unicode 转换为备用值。

注: 其他读写方法不使用 CharacterSet。

- ReadChar 和 WriteChar 在不进行转换的情况下在消息缓冲区中读写 Unicode 字符。
- ReadUTF 和 WriteUTF 在应用程序中的 Unicode 字符串与消息缓冲区中以 2 字节长度字段为前缀的 UTF-8 字符串之间进行转换。
- Byte 方法在应用程序和消息缓冲区之间传输字节而不进行更改。

public byte[] CorrelationId {get; set;}

- 对于 MQQueue.Get 调用，这是要检索的消息的相关标识。队列管理器返回具有与消息描述符字段匹配的消息标识和相关标识的第一条消息。缺省值 MQC.MQCI_NONE 可帮助任何相关标识进行匹配。
- 对于 MQQueue.Put 调用，这是要设置的相关标识。

public int DataLength {get;}

要读取的消息数据的剩余字节数。

public int DataOffset {get; set;}

消息数据中的当前光标位置。读写在当前位置生效。

public int Encoding {get; set;}

用于应用程序消息数据中的数字值的表示。编码适用于二进制，压缩十进制和浮点数据。这些数字格式的读写方法的行为会相应改变。通过从这三个部分中的每个部分添加一个值来构造编码字段的值。或者，使用按位 OR 运算符来构造值，以组合三个部分中的每个部分的值。

1. BINARY INTEGER

MQC.MQENC_INTEGER_NORMAL

大尾数法整数。

MQC.MQENC_INTEGER_REVERSED

在 Intel 体系结构中使用的小尾数法整数。

2. 压缩十进制

MQC.MQENC_DECIMAL_NORMAL

大尾数法压缩十进制，由 z/OS 使用。

MQC.MQENC_DECIMAL_REVERSED

小尾数法压缩十进制。

3. 浮点数

MQC.MQENC_FLOAT_IEEE_NORMAL

大尾数法 IEEE 浮点数。

MQC.MQENC_FLOAT_IEEE_REVERSED

小尾数法 IEEE 浮点运算，如使用的英特尔架构。

MQC.MQENC_FLOAT_S390

z/OS 格式浮点。

缺省值为：

```
MQC.MQENC_INTEGER_REVERSED |
MQC.MQENC_DECIMAL_REVERSED |
MQC.MQENC_FLOAT_IEEE_REVERSED
```

缺省设置导致 WriteInt 写入小尾数法整数， ReadInt 读取小尾数法整数。如果改为设置标志 MQC.MQENC_INTEGER_NORMAL 标志，那么 WriteInt 将写入大尾数法整数，而 ReadInt 将读取大尾数法整数。

注：从 IEEE 格式浮点转换为 zSeries 格式浮点时，可能会丢失精度。

public int Expiry {get; set;}

由放置消息的应用程序设置的到期时间 (以十分之一秒为单位)。在经过消息的到期时间后，队列管理器可将其废弃。如果消息指定了其中一个 MQC.MQRO_EXPIRATION 标志，那么将在废弃消息时生成报告。缺省值为 MQC.MQEI_UNLIMITED，表示消息永不到期。

public int Feedback {get; set;}

将反馈与类型为 MQC.MQMT_REPORT 的消息配合使用，以指示报告的性质。系统定义了以下反馈代码：

- 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

还可以使用 MQC.MQFB_APPL_FIRST 到 MQC.MQFB_APPL_LAST 范围内的应用程序定义的反馈值。此字段的缺省值为 MQC.MQFB_NONE，表示未提供任何反馈。

public string Format {get; set;}

消息发送方使用的格式名，用于向接收方指示消息中数据的性质。您可以使用自己的格式名称，但以字母 MQ 开头的名称具有队列管理器定义的含义。队列管理器内置格式为：

MQC.MQFMT_ADMIN

命令服务器请求/应答消息。

MQC.MQFMT_COMMAND_1

输入 1 命令应答消息。

MQC.MQFMT_COMMAND_2

输入 2 命令应答消息。

MQC.MQFMT_DEAD_LETTER_HEADER

死信头。

MQC.MQFMT_EVENT

事件消息。

MQC.MQFMT_NONE

无格式名称。

MQC.MQFMT_PCF

可编程命令格式的用户定义消息。

MQC.MQFMT_STRING

完全由字符组成的消息。

MQC.MQFMT_TRIGGER

触发器消息

MQC.MQFMT_XMIT_Q_HEADER

传输队列头。

缺省值为 MQC.MQFMT_NONE。

public byte[] GroupId {get; set;}

用于标识物理消息所属的消息组的字节字符串。缺省值为 MQC.MQGI_NONE。

public int MessageFlags {get; set;}

控制消息的分段和状态的标志。

public byte[] MessageId {get; set;}

对于 MQQueue.Get 调用，此字段指定要检索的消息的消息标识。通常，队列管理器返回具有与消息描述符字段匹配的消息标识和相关标识的第一条消息。允许任何消息标识使用特殊值 MQC.MQMI_NONE 进行匹配。

对于 MQQueue.Put 调用，此字段指定要使用的消息标识。如果指定了 MQC.MQMI_NONE，那么队列管理器将在放入消息时生成唯一消息标识。在 put 后更新此成员变量的值，以指示使用的消息标识。缺省值为 MQC.MQMI_NONE。

public int MessageLength {get;}

MQMessage 对象中消息数据的字节数。

public int MessageSequenceNumber {get; set;}

组中逻辑消息的序号。

public int MessageType {get; set;}

指示消息的类型。系统当前定义了以下值：

- MQC.MQMT_DATAGRAM
- MQC.MQMT_REPLY
- MQC.MQMT_REPORT
- MQC.MQMT_REQUEST

还可以使用应用程序定义的值，范围为 MQC.MQMT_APPL_FIRST 到 MQC.MQMT_APPL_LAST。此字段的缺省值为 MQC.MQMT_DATAGRAM。

public int Offset {get; set;}

在分段消息中，物理消息中的数据与逻辑消息开头的偏移量。

public int OriginalLength {get; set;}

分段消息的原始长度。

public int Persistence {get; set;}

消息持久性。已定义下列值：

- MQC.MQPER_NOT_PERSISTENT

如果在可重新连接的客户机中设置此选项，那么当连接成功时，会将 MQRC_NONE 原因码返回到应用程序。

- MQC.MQPER_PERSISTENT

如果在可重新连接的客户机中设置此选项，那么在连接成功后，会将 MQRC_CALL_INTERRUPTED 原因码返回到应用程序。

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

缺省值为 MQC.MQPER_PERSISTENCE_AS_Q_DEF，这将从目标队列的缺省持久性属性获取消息的持久性。

public int Priority {get; set;}

消息优先级。还可以在出站消息中设置特殊值 MQC.MQPRI_PRIORITY_AS_Q_DEF。然后，将从目标队列的缺省优先级属性中获取消息的优先级。缺省值为 MQC.MQPRI_PRIORITY_AS_Q_DEF。

public int PropertyValidation {get; set;}

指定在设置消息的属性时是否进行属性验证。可能的值为：

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

缺省值为 MQCMHO_DEFAULT_VALIDATION。

public string PutApplicationName {get; set;}

放置消息的应用程序的名称。缺省值为 ""。

public int PutApplicationType {get; set;}

放置消息的应用程序类型。PutApplication 类型 可以是系统定义的值或用户定义的值。系统定义了以下值：

- 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

缺省值为 MQC.MQAT_NO_CONTEXT，指示消息中不存在任何上下文信息。

public DateTime PutDateTime {get; set;}

放入消息的时间和日期。

public string ReplyToQueueManagerName {get; set;}

要发送应答或报告消息的队列管理器的名称。缺省值为 ""，队列管理器提供 ReplyToQueueManagerName。

public string ReplyToQueueName {get; set;}

发出消息获取请求的应用程序向其发送 MQC.MQMT_REPLY 和 MQC.MQMT_REPORT 消息的消息队列的名称。缺省 ReplyToQueueName 为 ""。

public int Report {get; set;}

使用 **报告** 来指定有关报告和应答消息的选项：

- 是否需要报告。
- 是否将应用程序消息数据包括在报告中。
- 如何在报告或应答中设置消息和相关标识。

可以请求这四种报告类型的任意组合：

- 指定四种报告类型的任意组合。根据应用程序消息数据是否包含在报告消息中，为每种报告类型选择三个选项中的任何一个。

1. 到达时确认

- MQC.MQRO_COA
- MQC.MQRO_COA_WITH_DATA

– MQC.MQRO_COA_WITH_FULL_DATA**

2. 传送时确认

– MQC.MQRO_COD

– MQC.MQRO_COD_WITH_DATA

– MQC.MQRO_COD_WITH_FULL_DATA**

3. 异常

– MQC.MQRO_EXCEPTION

– MQC.MQRO_EXCEPTION_WITH_DATA

– MQC.MQRO_EXCEPTION_WITH_FULL_DATA**

4. 到期

– MQC.MQRO_EXPIRATION

– MQC.MQRO_EXPIRATION_WITH_DATA

– MQC.MQRO_EXPIRATION_WITH_FULL_DATA**

注: z/OS 队列管理器不支持在列表中使用 ** 标记的值。如果应用程序可能访问 z/OS 队列管理器 (无论应用程序在哪个平台上运行), 请不要使用它们。

- 指定下列其中一项以控制如何为报告或应答消息生成消息标识:

– MQC.MQRO_NEW_MSG_ID

– MQC.MQRO_PASS_MSG_ID

- 指定下列其中一项以控制如何设置报告或应答消息的相关标识:

– MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID

– MQC.MQRO_PASS_CORREL_ID

- 指定下列其中一项以在无法将原始消息传递到目标队列时控制其处置:

– MQC.MQRO_DEAD_LETTER_Q

– MQC.MQRO_DISCARD_MSG**

- 如果未指定任何报告选项, 那么缺省值为:

```
MQC.MQRO_NEW_MSG_ID |
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |
MQC.MQRO_DEAD_LETTER_Q
```

- 您可以指定以下一项或两项以请求接收应用程序发送肯定操作或否定操作报告消息。

– MQC.MQRO_PAN

– MQC.MQRO_NAN

public int TotalMessageLength {get;}

消息中存储在从中接收此消息的消息队列上的总字节数。

public string UserId {get; set;}

UserId 是消息的身份上下文的一部分。队列管理器通常提供值。如果您有权设置身份上下文, 那么可以覆盖该值。

public int Version {get; set;}

正在使用的 MQMD 结构的版本。

Read 和 Write 消息方法

Read 和 Write 方法与 .NET System.IO 名称空间中的 BinaryReader 和 BinaryWriter 类的成员执行相同的功能。请参阅 MSDN 以获取完整语言语法和用法示例。从消息缓冲区中的当前位置读取或写入的方法。它们将当前位置向前移动读取或写入的字节数。

注: 如果消息数据包含 MQRFH 或 MQRFH2 头, 那么必须使用 ReadBytes 方法来读取数据。

- 所有方法都将抛出 `IOException`。
- `ReadFully` 方法会自动调整目标 `byte` 或 `sbyte` 数组的大小以准确拟合消息。还会调整空数组的大小。
- `Read` 方法抛出 `EndOfStreamException`。
- `WriteDecimal` 方法抛出 `MQException`。
- `ReadString`, `ReadLine` 和 `WriteString` 方法在 Unicode 和消息字符集之间转换; 请参阅 `CharacterSet`。
- `Decimal` 方法根据 编码 的值来读写以大尾数法, `MQC.MQENC_DECIMAL_NORMAL` 或小尾数法 `MQC.MQENC_DECIMAL_REVERSE` 格式编码的压缩十进制数字。十进制范围和相应的 .NET 类型如下所示:

Decimal2/short

-999 到 999

Decimal4/int

-9999999 到 9999999

Decimal8/long

-9999999999999999 到 9999999999999999

- `Double` 和 `Float` 方法根据 编码 的值来读写以 IEE 大尾数法和小尾数法格式 (`MQC.MQENC_FLOAT_IEEE_NORMAL` 和 `MQC.MQENC_FLOAT_IEEE_REVERSED`) 或 S/390 格式 (`MQC.MQENC_FLOAT_S390`) 编码的浮点值。
- `Int` 方法根据 编码 的值来读写以大尾数法, `MQC.MQENC_INTEGER_NORMAL` 或小尾数法 `MQC.MQENC_INTEGER_REVERSED` 格式编码的整数值。除添加无符号的 2 字节整数类型外, 所有整数都是有符号的。整数大小以及 .NET 和 WebSphere MQ 类型如下所示:

2 字节

`short`, `Int2`, `ushort`, `UInt2`

4 字节

`int`, `Int4`

8 字节

`long`, `Int8`

- `WriteObject` 将对象的类, 其非瞬态和非静态字段的值及其超类型的字段传输到消息缓冲区。
- `ReadObject` 根据对象的类, 类的特征符, 其非瞬态字段和非静态字段的值以及其超类型的字段创建对象。

| 表 608: 读和写消息方法 | |
|----------------|--|
| 目标类型 | 方法特征符 |
| Boolean | <pre>public bool ReadBoolean(); public void WriteBoolean(bool value);</pre> |
| Byte | <pre>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)</pre> |

表 608: 读和写消息方法 (继续)

| 目标类型 | 方法特征符 |
|-----------------|---|
| Bytes | <pre>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)</pre> |
| Decimal2 | <pre>public void WriteDecimal2(short value)</pre> |
| Decimal4 | <pre>public void WriteDecimal4(short value)</pre> |
| Decimal8 | <pre>public void WriteDecimal8(short value)</pre> |
| Double | <pre>public double ReadDouble() public void WriteDouble(double value)</pre> |
| Float | <pre>public float ReadFloat() public void WriteFloat(float value)</pre> |
| Int2 | <pre>public void WriteInt2(int value)</pre> |
| Int4 | <pre>public int readDecimal4() public int ReadInt() public int ReadInt4() public void WriteInt(int value) public void WriteInt4(int value)</pre> |
| Int8 | <pre>public void WriteInt8(long value)</pre> |
| Long | <pre>public long ReadDecimal8() public long ReadLong() public long ReadInt8() public void WriteLong(long value)</pre> |
| Object | <pre>public Object ReadObject() public void WriteObject(Object object)</pre> |

表 608: 读和写消息方法 (继续)

| 目标类型 | 方法特征符 |
|-----------------------|--|
| Short | <pre>public short ReadShort() public short ReadDecimal2() public short ReadInt2() public void WriteShort(int value)</pre> |
| string | <pre>public string ReadString(int length) public void WriteString(string string)</pre> |
| Unsigned Short | <pre>public ushort ReadUnsignedShort() public ushort ReadUInt2()</pre> |
| Unicode | <pre>public string ReadLine() public char ReadChar() public void WriteChar(int value) public void WriteChars(string string)</pre> |
| UTF | <pre>public string ReadUTF() public void WriteUTF(string string)</pre> |

缓冲区方法

public void ClearMessage();

抛出 `IOException`。

废弃消息缓冲区中的任何数据，并将数据偏移量设置回零。

public void ResizeBuffer(int size)

抛出 `IOException`。

向 `MQMessage` 对象提供有关后续获取操作可能需要的缓冲区大小的提示。如果消息当前包含消息数据，并且新大小小于当前大小，那么将截断消息数据。

public void Seek(int pos)

抛出 `IOException`，`ArgumentOutOfRangeException` 和 `ArgumentException`。

将光标移动到 `pos` 给出的消息缓冲区中的绝对位置。后续读和写操作在缓冲区中的此位置。

public int SkipBytes(int i)

抛出 `IOException` 和 `EndOfStreamException`。

在消息缓冲区中向前移动 `n` 个字节，并返回 `n`，即跳过的字节数。

`SkipBytes` 方法阻塞，直到发生下列其中一个事件：

- 跳过所有字节
- 检测到消息缓冲区结束
- 抛出了异常

属性方法

public void DeleteProperty(string name);

抛出 MQException。

从消息中删除具有指定名称的属性。

name

要删除的属性的名称。

public System.Collections.IEnumerator GetPropertyNames(string name)

抛出 MQException。

返回与指定名称匹配的所有属性名的 IEnumerator。可以在名称末尾使用百分号 '%' 作为通配符，以过滤消息的属性，在零个或多个字符 (包括句点) 上进行匹配。

name

要匹配的属性的名称。

- 所有 SetProperty 和 GetProperty 方法都抛出 MQException

| 类型 | 方法特征符 |
|----------------|---|
| Boolean | <pre>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);</pre> |
| Byte | <pre>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);</pre> |
| Bytes | <pre>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);</pre> |
| Double | <pre>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);</pre> |
| Float | <pre>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);</pre> |
| Int2 | <pre>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);</pre> |

表 609: SetProperty 和 GetProperty 方法 (继续)

| 类型 | 方法特征符 |
|---------------|---|
| Int4 | <pre>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);</pre> |
| Int8 | <pre>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);</pre> |
| Long | <pre>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);</pre> |
| Object | <pre>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);</pre> |
| Short | <pre>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);</pre> |
| string | <pre>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);</pre> |

构造函数

public MQMessage();

使用缺省消息描述符信息和空消息缓冲区创建 MQMessage 对象。

MQProcess .NET 类

使用 MQProcess 来查询 WebSphere MQ 进程的属性。使用构造函数或 MQQueueManager AccessProcess 方法创建 MQProcess 对象。

类

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQProcess

```

```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```

- [第 1143 页的『属性』](#)
- [第 1143 页的『构造函数』](#)

属性

获取属性时抛出 `MQException` 的测试。

```
public string ApplicationId {get;}
```

获取用于标识要启动的应用程序的字符串。ApplicationId 由触发器监视器应用程序使用。ApplicationId 作为触发器消息的一部分发送到启动队列。

缺省值为空。

```
public int ApplicationType {get;}
```

标识要由触发器监视器应用程序启动的进程的类型。定义了标准类型，但可以使用其他类型：

- MQAT_AIX
- MQAT_CICS
- MQAT_IMS
- MQAT_MVS
- MQAT_NATIVE
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

缺省值为 MQAT_NATIVE。

```
public string EnvironmentData {get;}
```

获取有关要启动的应用程序环境的信息。

缺省值为空。

```
public string UserData {get;}
```

获取用户提供的有关要启动的应用程序的信息。

缺省值为空。

构造函数

```
public MQProcess(MQQueueManager queueManager, string processName, int openOptions);
```

```
public MQProcess(MQQueueManager qMgr, string processName, int openOptions, string queueManagerName, string alternateUserId);
```

抛出 `MQException`。

访问队列管理器 *qMgr* 上的 WebSphere MQ 进程以查询进程属性。

qMgr

要访问的队列管理器。

processName

要打开的进程的名称。

openOptions

用于控制进程打开的选项。可以添加或使用按位 OR 组合的有效选项包括:

- MQC.MQOO_FAIL_IF QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName

在其中定义进程的队列管理器的名称。如果队列管理器与进程正在访问的队列管理器相同，那么可以保留空白或空的队列管理器名称。

alternateUserId

如果在 *openOptions* 参数中指定了 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 将指定用于检查操作授权的备用用户标识。如果未指定 MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 可以为空或为空。

如果未指定 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么将使用缺省用户权限来连接到队列管理器。

```
public MQProcess MQQueueManager.AccessProcess(string processName, int  
openOptions);  
public MQProcess MQQueueManager.AccessProcess(string processName, int  
openOptions, string queueManagerName, string alternateUserId);
```

抛出 MQException。

访问此队列管理器上的 WebSphere MQ 进程以查询进程属性。

processName

要打开的进程的名称。

openOptions

用于控制进程打开的选项。可以添加或使用按位 OR 组合的有效选项包括:

- MQC.MQOO_FAIL_IF QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName

在其中定义进程的队列管理器的名称。如果队列管理器与进程正在访问的队列管理器相同，那么可以保留空白或空的队列管理器名称。

alternateUserId

如果在 *openOptions* 参数中指定了 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 将指定用于检查操作授权的备用用户标识。如果未指定 MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 可以为空或为空。

如果未指定 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么将使用缺省用户权限来连接到队列管理器。

MQPropertyDescriptor .NET 类

将 MQPropertyDescriptor 用作 MQMessage GetProperty 和 SetProperty 方法的参数。MQPropertyDescriptor 描述了 MQMessage 属性。

类

```
System.Object
└─ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```

- [第 1145 页的『属性』](#)
- [第 1146 页的『构造函数』](#)

属性

获取属性时抛出 MQException 的测试。

```
public int Context {get; set;}
```

属性所属的消息上下文。可能的值为：

MQC.MQPD_NO_CONTEXT

该属性未与消息上下文关联。

MQC.MQPD_USER_CONTEXT

该属性与用户上下文相关联。

如果用户已获得授权，那么检索消息时将保存与用户上下文相关联的属性。引用已保存上下文的后续 Put 方法可以将该属性传递到新消息中。

```
public int CopyOptions {get; set;}
```

CopyOptions 描述可将属性复制到的消息类型。

当队列管理器接收到包含 WebSphere MQ 定义的属性 (队列管理器可识别为不正确) 的消息时，队列管理器将更正 CopyOptions 字段的值。

可以指定以下选项的任意组合。通过添加值或使用按位 OR 来组合选项。

MQC.MQCOPY_ALL

此属性将复制到所有类型的后续消息中。

MQC.MQCOPY_FORWARD

该属性将复制到要转发的消息中。

MQC.MQCOPY_PUBLISH

在发布消息时，会将该属性复制到订户接收到的消息中。

MQC.MQCOPY_REPLY

该属性将复制到应答消息中。

MQC.MQCOPY_REPORT

该属性将复制到报告消息中。

MQC.MQCOPY_DEFAULT

该值指示未指定任何其他复制选项。该属性与后续消息之间不存在任何关系。始终针对消息描述符属性返回 MQC.MQCOPY_DEFAULT。

MQC.MQCOPY_NONE

与 MQC.MQCOPY_DEFAULT 相同

```
public int Options { set; }
```

选项 缺省为 CMQC.MQPD_NONE。不能设置任何其他值。

```
public int Support { get; set; }
```

设置 `Support` 以指定 WebSphere MQ 定义的消息属性所需的支持级别。对所有其他属性的支持是可选的。可以指定下列任何值，也可以不指定任何值

MQC.MQPD_SUPPORT_OPTIONAL

队列管理器接受该属性，即使该属性不受支持也是如此。可以废弃该属性以使消息流向不支持消息属性的队列管理器。此值还会分配给未定义 WebSphere MQ 的属性。

MQC.MQPD_SUPPORT_REQUIRED

需要对该属性的支持。如果将消息放入不支持 WebSphere MQ 定义的属性的队列管理器，那么该方法将失败。它返回完成代码 `MQC.MQCC_FAILED` 和原因码 `MQC.MQRC_UNSUPPORTED_PROPERTY`。

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

如果消息以本地队列为目标，那么需要对该属性的支持。如果将消息放入不支持 WebSphere MQ 定义的属性的队列管理器上的本地队列，那么该方法将失败。它返回完成代码 `MQC.MQCC_FAILED` 和原因码 `MQC.MQRC_UNSUPPORTED_PROPERTY`。

如果将消息放入远程队列管理器，那么不会进行任何检查。

构造函数

```
PropertyDescriptor();
```

创建属性描述符。

MQPutMessageOptions.NET 类

使用 `MQPutMessageOptions` 来指定如何发送消息。它会修改 `MQDestination.Put` 的行为。

类

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQPutMessageOptions
```

```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

• [第 1146 页的『属性』](#) [第 1148 页的『构造函数』](#)

属性

获取属性时抛出 `MQException` 的测试。

注: 此类中某些可用选项的行为取决于使用这些选项的环境。这些元素标有星号 *。

```
public MQQueue ContextReference {get; set;}
```

如果 `options` 字段包含 `MQC.MQPMO_PASS_IDENTITY_CONTEXT` 或 `MQC.MQPMO_PASS_ALL_CONTEXT`，请设置此字段以引用要从中获取上下文信息的 `MQQueue`。

此字段的初始值为空。

```
public int InvalidDestCount {get;}*
```

通常，用于分发列表，`InvalidDestCount` 指示无法发送到分发列表中的队列的消息数。计数包括未能打开的队列以及已成功打开但放置操作失败的队列。

.NET 不支持分发列表，但在打开单个队列时设置了 `InvalidDestCount`。

public int KnownDestCount {get;} *

通常用于分发列表，KnownDestCount 指示当前调用已成功发送到解析为本地队列的队列的消息数。
.NET 不支持分发列表，但在打开单个队列时设置了 InvalidDestCount。

public int Options {get; set;}

用于控制 MQDestination.put 和 MQQueueManager.put 的操作的选项。可以指定以下任何值，也可以不指定任何值。如果需要多个选项，那么可以使用按位 OR 运算符来添加或组合值。

MQC.MQPMO_ASYNC_RESPONSE

此选项会导致 MQDestination.put 调用以异步方式进行，并带有一些响应数据。

MQC.MQPMO_DEFAULT_CONTEXT

将缺省上下文与消息关联。

MQC.MQPMO_FAIL_IF QUIESCING

如果队列管理器正在停顿，那么失败。

MQC.MQPMO_LOGICAL_ORDER*

将消息组中的逻辑消息和段置于其逻辑顺序中。

如果在可重新连接的客户机中使用 MQPMO_LOGICAL_ORDER 选项，那么会将 MQRC_RECONNECT_INCOMPATIBLE 原因码返回到应用程序。

MQC.MQPMO_NEW_CORREL_ID*

为每条已发送的消息生成新的相关标识。

MQC.MQPMO_NEW_MSG_ID*

为每个已发送的消息生成新的消息标识。

MQC.MQPMO_NONE

未指定任何选项。请勿与其他选项一起使用。

MQC.MQPMO_NO_CONTEXT

没有要与消息关联的上下文。

MQC.MQPMO_NO_SYNCPOINT

在没有同步点控制的情况下放入消息。如果未指定同步点控制选项，那么将采用缺省值 "无同步点"。

MQC.MQPMO_PASS_ALL_CONTEXT

从输入队列句柄传递所有上下文。

MQC.MQPMO_PASS_IDENTITY_CONTEXT

从输入队列句柄传递身份上下文。

MQC.MQPMO_RESPONSE_AS_Q_DEF

对于 MQDestination.put 调用，此选项采用来自队列的 DEFPRESP 属性的 put 响应类型。

对于 MQQueueManager.put 调用，此选项会使调用同步进行。

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF 是 MQC.MQPMO_RESPONSE_AS_Q_DEF 的同义词，用于主题对象。

MQC.MQPMO_RETAIN

要发送的发布将由队列管理器保留。如果使用此选项并且无法保留发布，那么将不会发布消息，并且调用将失败并返回 MQC.MQRC_PUT_NOT_RETAINED。

在发布此出版物之后，通过调用 MQSubscription.RequestPublicationUpdate 方法来请求此出版物的副本。已保存的发布将发送到创建预订的应用程序，而不设置 MQC.MQSO_NEW_PUBLICATIONS_ONLY 选项。在接收到发布时，请检查该发布的 MQIsRetained 消息属性，以了解它是否是保留发布。

订户请求保留发布时，所使用的预订可能在主题字符串中包含通配符。如果主题树中有多个与预订匹配的保留发布，那么将全部发送这些发布。

MQC.MQPMO_SET_ALL_CONTEXT

设置应用程序中的所有上下文。

MQC.MQPMO_SET_IDENTITY_CONTEXT

从应用程序设置身份上下文。

MQC.MQPMO_SYNC_RESPONSE

此选项将使 MQDestination.put 或 MQQueueManager.put 调用与完整响应数据同步进行。

MQC.MQPMO_SUPPRESS_REPLYTO

在出版物的 ReplyToQueueName 和 ReplyToQueueManagerName 字段中填写的任何信息都不会传递给订户。如果将此选项与需要 ReplyToQueueName 的报告选项结合使用，那么调用将失败并返回 MQC.MQRC_MISSING_REPLY_TO_Q。

MQC.MQPMO_SYNCPOINT

放置具有同步点控制的消息。在落实工作单元之前，该消息在工作单元外部不可见。如果工作单元已回退，那么将删除该消息。

public int RecordFields {get; set;} *

有关分发列表的信息。分发列表在 .NET 中不支持。

public string ResolvedQueueManagerName {get;}

队列管理器将输出字段设置为拥有由远程队列名称指定的队列的队列管理器的名称。

ResolvedQueueManagerName 可能与从中访问队列的队列管理器的名称不同 (如果该队列是远程队列)。

仅当对象是单个队列时，才会返回非空白值。如果对象是分发列表或主题，那么返回的值未定义。

public string ResolvedQueueName {get;}

由队列管理器设置为放置消息的队列的名称的输出字段。ResolvedQueue 名称可能与用于打开队列的名称不同 (如果打开的队列是别名或模型队列)。

仅当对象是单个队列时，才会返回非空白值。如果对象是分发列表或主题，那么返回的值未定义。

public int UnknownDestCount {get;} *

通常用于分发列表，UnknownDestCount 是队列管理器设置的输出字段。它报告当前调用已成功发送到解析为远程队列的队列的消息数。

.NET 不支持分发列表，但在打开单个队列时设置了 InvalidDestCount。

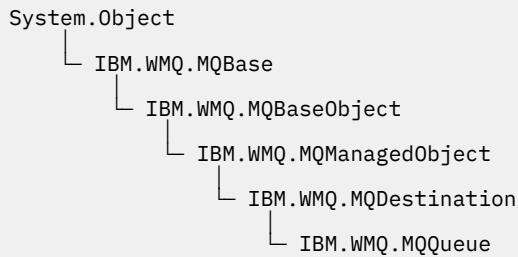
构造函数**public MQPutMessageOptions();**

构造没有设置选项的新 MQPutMessageOptions 对象，以及空白 ResolvedQueueName 和 ResolvedQueueManagerName。

MQQueue .NET 类

使用 MQQueue 来发送和接收消息以及 WebSphere MQ 队列的查询属性。使用构造函数或 MQQueueManager.AccessProcess 方法创建 MQQueue 对象。

类



```
public class IBM.WMQ.MQQueue extends IBM.WMQ.MQDestination;
```

- [第 1149 页的『属性』](#)
- [第 1151 页的『方法』](#)
- [第 1153 页的『构造函数』](#)

属性

获取属性时抛出 `MQException` 的测试。

public int ClusterWorkLoadPriority {get;}

指定队列的优先级。此参数仅对本地，远程和别名队列有效。

public int ClusterWorkLoadRank {get;}

指定队列的列组。此参数仅对本地，远程和别名队列有效。

public int ClusterWorkLoadUseQ {get;}

指定当目标队列具有本地实例和至少一个远程集群实例时 `MQPUT` 操作的行为。如果 `MQPUT` 源自集群通道，那么此参数不适用。此参数仅对本地队列有效。

public DateTime CreationDateTime {get;}

创建此队列的日期和时间。

public int CurrentDepth {get;}

获取当前在队列中的消息数。在 `put` 调用期间以及在回退 `get` 调用期间，此值递增。在非浏览获取期间以及在回退 `put` 调用期间，将对其进行递减。

public int DefinitionType {get;}

队列的定义方式。可能的值为：

- `MQC.MQQDT_PREDEFINED`
- `MQC.MQQDT_PERMANENT_DYNAMIC`
- `MQC.MQQDT_TEMPORARY_DYNAMIC`

public int InhibitGet {get; set;}

控制您是可以获取此队列上的消息还是获取此主题的消息。可能的值为：

- `MQC.MQQA_GET_INHIBITED`
- `MQC.MQQA_GET_ALLOWED`

public int InhibitPut {get; set;}

控制是可以将消息放入此队列还是针对此主题。可能的值为：

- `MQQA_PUT_INHIBITED`
- `MQQA_PUT_ALLOWED`

public int MaximumDepth {get;}

队列中可同时存在的最大消息数。尝试将消息放入已包含此大量消息的队列失败，原因码为 `MQC.MQRC_Q_FULL`。

public int MaximumMessageLength {get;}

此队列中每条消息中可存在的应用程序数据的最大长度。尝试放置大于此值的消息失败，原因码为 `MQC.MQRC_MSG_TOO_BIG_FOR_Q`。

public int NonPersistentMessageClass {get;}

放入此队列的非持久消息的可靠性级别。

public int OpenInputCount {get;}

当前对从队列中除去消息有效的句柄数。OpenInputCount 是本地队列管理器已知的有效输入句柄的总数，而不仅仅是应用程序创建的句柄。

public int OpenOutputCount {get;}

当前对将消息添加到队列有效的句柄数。OpenOutputCount 是本地队列管理器已知的有效输出句柄总数，而不仅仅是应用程序创建的句柄。

public int QueueAccounting {get;}

指定是否可以对队列启用记帐信息收集。

public int QueueMonitoring {get;}

指定是否可以对队列启用监视。

public int QueueStatistics {get;}

指定是否可以对队列启用统计信息收集。

public int QueueType {get;}

具有下列其中一个值的此队列的类型:

- MQC.MQQT_ALIAS
- MQC.MQQT_LOCAL
- MQC.MQQT_REMOTE
- MQC.MQQT_CLUSTER

public int Shareability {get;}

是否可以多次打开队列以进行输入。可能的值为:

- MQC.MQQA_SHAREABLE
- MQC.MQQA_NOT_SHAREABLE

public string TPIPE {get;}

用于使用 WebSphere MQ IMS 网桥与 OTMA 通信的 TPIPE 名称。

public int TriggerControl {get; set;}

是否将触发器消息写入启动队列，以启动应用程序来为队列提供服务。可能的值为:

- MQC.MQTC_OFF
- MQC.MQTC_ON

public string TriggerData {get; set;}

队列管理器插入到触发器消息中的自由格式数据。当到达此队列的消息导致触发器消息写入启动队列时，它会插入 TriggerData。字符串的最大允许长度由 MQC.MQ_TRIGGER_DATA_LENGTH 给出。

public int TriggerDepth {get; set;}

当触发器类型设置为 MQC.MQTT_DEPTH 时，在写入触发器消息之前必须在队列上的消息数。

public int TriggerMessagePriority {get; set;}

消息不参与触发器消息生成的消息优先级。即，队列管理器在决定是否生成触发器时忽略这些消息。值为零将导致所有消息生成触发器消息。

public int TriggerType {get; set;}

由于消息到达此队列而写入触发器消息的条件。可能的值为:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT EVERY
- MQC.MQTT_DEPTH

方法

```
public void Get(MQMessage message);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int  
MaxMsgSize);
```

抛出 MQException。

从队列获取消息。

如果 get 失败，那么 MQMessage 对象保持不变。如果成功，那么 MQMessage 的消息描述符和消息数据部分将替换为来自入局消息的消息描述符和消息数据。

从特定 MQQueueManager 对 WebSphere MQ 的所有调用都是同步的。因此，如果执行获取并等待，那么将阻止使用相同 MQQueueManager 的所有其他线程进行进一步的 WebSphere MQ 调用，直到完成 Get 调用为止。如果需要多个线程来同时访问 WebSphere MQ，那么每个线程都必须创建自己的 MQQueueManager 对象。

message

包含消息描述符和返回的消息数据。消息描述符中的某些字段是输入参数。确保根据需要设置 MessageId 和 CorrelationId 输入参数非常重要。

对于在 MQGM_SYNCPOINT 下接收到的消息，可重新连接的客户机在成功重新连接后返回原因码 MQRC_BACKED_OUT。

getMessageOptions

控制获取操作的选项。

使用选项 MQC.MQMO_CONVERT 可能会导致在从单字节字符代码转换为双字节代码时发生异常，原因码为 MQC.MQRC_CONVERTED_STRING_TOO_BIG。在这种情况下，会将消息复制到缓冲区中而不进行转换。

如果未指定 getMessageOptions，那么使用的消息选项为 MQGMO_NOWAIT。

如果在可重新连接的客户机中使用 MQGMO_LOGICAL_ORDER 选项，那么将返回 MQRC_RECONNECT_INCOMPATIBLE 原因码。

MaxMsgSize

此消息对象要接收的最大消息。如果队列上的消息大于此大小，那么将发生以下两种情况之一：

- 如果在 MQGetMessageOptions 对象中设置了 MQGMO_ACCEPT_TRUNCATED_MSG 标志，那么将使用尽可能多的消息数据来填充消息。抛出异常时带有 MQCC_WARNING 完成代码和 MQRC_TRUNCATED_MSG_ACCEPTED 原因码。
- 如果未设置 MQGMO_ACCEPT_TRUNCATED_MSG 标志，那么消息将保留在队列中。抛出异常时带有 MQCC_WARNING 完成代码和 MQRC_TRUNCATED_MSG_FAILED 原因码。

如果未指定 MaxMsgSize，那么将检索整个消息。

```
public void Put(MQMessage message);  
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

抛出 MQException。

将消息放入队列。

完成 Put 调用后对 MQMessage 对象的修改不会影响 WebSphere MQ 队列或发布主题上的实际消息。

Put 会更新 MQMessage 对象的 MessageId 和 CorrelationId 属性，并且不会清除消息数据。进一步的 Put 或 Get 调用引用 MQMessage 对象中的更新信息。例如，在以下代码片段中，第一条消息包含 a 和第二条 ab。

```
msg.WriteString("a");  
q.Put(msg, pmo);  
msg.WriteString("b");  
q.Put(msg, pmo);
```

message

包含消息描述符数据和要发送的消息的 `MQMessage` 对象。由于此方法，可以更改消息描述符。在此方法完成后，消息描述符中的值是已放入队列或已发布到主题的值。

以下原因码将返回到可重新连接的客户机：

- `MQRC_CALL_INTERRUPTED` : 如果在对持久消息运行 `Put` 调用时连接中断，并且重新连接成功。
- `MQRC_NONE` (如果在对非持久消息运行 `Put` 调用时连接成功) (请参阅 [应用程序恢复](#))。

putMessageOptions

用于控制 `put` 操作的选项。

如果未指定 `putMessageOptions`，那么将使用 `MQPutMessageOptions` 的缺省实例。

如果在可重新连接的客户机中使用 `MQPMO_LOGICAL_ORDER` 选项，那么将返回 `MQRC_RECONNECT_INCOMPATIBLE` 原因码。

注：为了实现简单性和性能，如果要将单个消息放入队列，请使用 `MQQueueManager.Put` 对象。您应该具有此对象的 `MQQueue` 对象。

```
public void PutForwardMessage(MQMessage message);  
public void PutForwardMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions);
```

抛出 `MQException`

将正在转发的消息放入队列中，其中 `message` 是原始消息。

message

包含消息描述符数据和要发送的消息的 `MQMessage` 对象。由于此方法，可以更改消息描述符。在此方法完成后，消息描述符中的值是已放入队列或已发布到主题的值。

以下原因码将返回到可重新连接的客户机：

- `MQRC_CALL_INTERRUPTED` : 如果在对持久消息运行 `Put` 调用时连接中断，并且重新连接成功。
- `MQRC_NONE` (如果在对非持久消息运行 `Put` 调用时连接成功) (请参阅 [应用程序恢复](#))。

putMessageOptions

用于控制 `put` 操作的选项。

如果未指定 `putMessageOptions`，那么将使用 `MQPutMessageOptions` 的缺省实例。

如果在可重新连接的客户机中使用 `MQPMO_LOGICAL_ORDER` 选项，那么将返回 `MQRC_RECONNECT_INCOMPATIBLE` 原因码。

```
public void PutReplyMessage(MQMessage message)  
public void PutReplyMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

抛出 `MQException`。

将应答消息放入队列中，其中 `message` 是原始消息。

message

包含消息描述符和返回的消息数据。消息描述符中的某些字段是输入参数。确保根据需要设置 `MessageId` 和 `CorrelationId` 输入参数非常重要。

对于在 `MQGM_SYNCPOINT` 下接收到的消息，可重新连接的客户机在成功重新连接后返回原因码 `MQRC_BACKED_OUT`。

putMessageOptions

用于控制 `put` 操作的选项。

如果未指定 `putMessageOptions`，那么将使用 `MQPutMessageOptions` 的缺省实例。

如果在可重新连接的客户机中使用 `MQPMO_LOGICAL_ORDER` 选项，那么将返回 `MQRC_RECONNECT_INCOMPATIBLE` 原因码。


```
public void PutReportMessage(MQMessage message)  
public void PutReportMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

抛出 MQException。

将报告消息放入队列中，其中 *message* 是原始消息。

message

包含消息描述符和返回的消息数据。消息描述符中的某些字段是输入参数。确保根据需要设置 MessageId 和 CorrelationId 输入参数非常重要。

对于在 MQQM_SYNCPOINT 下接收到的消息，可重新连接的客户机在成功重新连接后返回原因码 MQRC_BACKED_OUT。

putMessageOptions

用于控制 put 操作的选项。

如果未指定 *putMessageOptions*，那么将使用 MQPutMessageOptions 的缺省实例。

如果在可重新连接的客户机中使用 MQPMO_LOGICAL_ORDER 选项，那么将返回 MQRC_RECONNECT_INCOMPATIBLE 原因码。

构造函数

```
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions);  
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

抛出 MQException。

访问此队列管理器上的队列。

您可以获取或浏览消息，放置消息，查询队列的属性或设置队列的属性。如果指定的队列是模型队列，那么将创建动态本地队列。查询生成的 MQQueue 对象的 name 属性以找出动态队列的名称。

queueName

要打开的队列的名称。

openOptions

用于控制队列打开的选项。

MQC.MQOO_ALTERNATE_USER_AUTHORITY

使用指定的用户标识进行验证。

MQC.MQOO_BIND_AS_QDEF

对队列使用缺省绑定。

MQC.MQOO_BIND_NOT_FIXED

请勿绑定到特定目标。

MQC.MQOO_BIND_ON_OPEN

打开队列时将句柄绑定到目标。

MQC.MQOO_BROWSE

打开以浏览消息。

MQC.MQOO_FAIL_IF QUIESCING

如果队列管理器正在停顿，那么失败。

MQC.MQOO_INPUT_AS_Q_DEF

打开以使用队列定义的缺省值获取消息。

MQC.MQOO_INPUT_SHARED

打开以获取具有共享访问权的消息。

MQC.MQOO_INPUT_EXCLUSIVE

打开以获取具有独占访问权的消息。

MQC.MQOO_INQUIRE

打开以进行查询-如果要查询属性，那么此选项是必需的。

MQC.MQOO_OUTPUT

打开以放置消息。

MQC.MQOO_PASS_ALL_CONTEXT

允许传递所有上下文。

MQC.MQOO_PASS_IDENTITY_CONTEXT

允许传递身份上下文。

MQC.MQOO_SAVE_ALL_CONTEXT

检索消息时保存上下文。

MQC.MQOO_SET

打开以设置属性-如果要设置属性，那么此选项是必需的。

MQC.MQOO_SET_ALL_CONTEXT

允许设置所有上下文。

MQC.MQOO_SET_IDENTITY_CONTEXT

允许设置身份上下文。

queueManagerName

定义队列的队列管理器的名称。完全为空白或空的名称表示 MQQueueManager 对象所连接的队列管理器。

dynamicQueueName

除非 *queueName* 指定模型队列的名称，否则将忽略 *dynamicQueueName*。如果这样做，那么 *dynamicQueueName* 将指定要创建的动态队列的名称。如果 *queueName* 指定模型队列的名称，那么空白或空名称无效。如果名称中的最后一个非空白字符是星号 *，那么队列管理器会将星号替换为字符串。这些字符保证为队列生成的名称在此队列管理器上是唯一的。

alternateUserId

如果在 *openOptions* 参数中指定了 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 指定用于检查打开权限的备用用户标识。如果未指定 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 可以留空或为空。

```
public MQQueue(MQQueueManager queueManager, string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

抛出 MQException。

访问 *queueManager* 上的队列。

您可以获取或浏览消息，放置消息，查询队列的属性或设置队列的属性。如果指定的队列是模型队列，那么将创建动态本地队列。查询生成的 MQQueue 对象的 *name* 属性以找出动态队列的名称。

queueManager

要访问队列的队列管理器。

queueName

要打开的队列的名称。

openOptions

用于控制队列打开的选项。

MQC.MQOO_ALTERNATE_USER_AUTHORITY

使用指定的用户标识进行验证。

MQC.MQOO_BIND_AS_QDEF

对队列使用缺省绑定。

MQC.MQOO_BIND_NOT_FIXED

请勿绑定到特定目标。

MQC.MQOO_BIND_ON_OPEN

打开队列时将句柄绑定到目标。

MQC.MQOO_BROWSE

打开以浏览消息。

MQC.MQOO_FAIL_IF QUIESCING

如果队列管理器正在停顿，那么失败。

MQC.MQOO_INPUT_AS_Q_DEF

打开以使用队列定义的缺省值获取消息。

MQC.MQOO_INPUT_SHARED

打开以获取具有共享访问权的消息。

MQC.MQOO_INPUT_EXCLUSIVE

打开以获取具有独占访问权的消息。

MQC.MQOO_INQUIRE

打开以进行查询-如果要查询属性，那么此选项是必需的。

MQC.MQOO_OUTPUT

打开以放置消息。

MQC.MQOO_PASS_ALL_CONTEXT

允许传递所有上下文。

MQC.MQOO_PASS_IDENTITY_CONTEXT

允许传递身份上下文。

MQC.MQOO_SAVE_ALL_CONTEXT

检索消息时保存上下文。

MQC.MQOO_SET

打开以设置属性-如果要设置属性，那么此选项是必需的。

MQC.MQOO_SET_ALL_CONTEXT

允许设置所有上下文。

MQC.MQOO_SET_IDENTITY_CONTEXT

允许设置身份上下文。

queueManagerName

定义队列的队列管理器的名称。完全为空白或空的名称表示 MQQueueManager 对象所连接的队列管理器。

dynamicQueueName

除非 *queueName* 指定模型队列的名称，否则将忽略 *dynamicQueueName*。如果这样做，那么 *dynamicQueueName* 将指定要创建的动态队列的名称。如果 *queueName* 指定模型队列的名称，那么空白或空名称无效。如果名称中的最后一个非空白字符是星号 *，那么队列管理器会将星号替换为字符串。这些字符保证为队列生成的名称在此队列管理器上是唯一的。

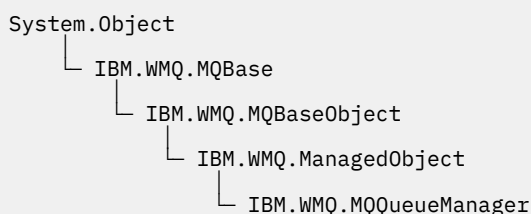
alternateUserId

如果在 *openOptions* 参数中指定了 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 指定用于检查打开权限的备用用户标识。如果未指定 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 可以留空或为空。

MQQueueManager .NET 类

使用 MQQueueManager 连接到队列管理器和访问队列管理器对象。它还控制事务。MQQueueManager 构造函数创建客户机或服务器连接。

类



```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```

- [第 1156 页的『属性』](#)
- [第 1159 页的『方法』](#)
- [第 1164 页的『构造函数』](#)

属性

获取属性时抛出 MQException 的测试。

```
public int AccountingConnOverride {get;}
```

应用程序是否可以覆盖 MQI 记帐和队列记帐值的设置。

```
public int AccountingInterval {get;}
```

写入中间记帐记录之前的时间长度 (以秒计)。

```
public int ActivityRecording {get;}
```

控制活动报告的生成。

```
public int AdoptNewMCACheck {get;}
```

指定检查哪些元素以确定在检测到新的进站通道时是否采用 MCA。要采用此参数, MCA 名称必须与活动 MCA 的名称匹配。

```
public int AdoptNewMCAInterval {get;}
```

新通道等待孤线程通道结束的时间量 (以秒计)。

```
public int AdoptNewMCAType {get;}
```

当检测到新的进站通道请求与 AdoptNewMCACheck 值匹配时, 是否采用 (重新启动) 孤立 MCA 实例。

```
public int BridgeEvent {get;}
```

是否生成 IMS 网桥事件。

```
public int ChannelEvent {get;}
```

是否生成通道事件。

```
public int ChannelInitiatorControl {get;}
```

在队列管理器启动时是否自动启动通道启动程序。

```
public int ChannelInitiatorAdapters {get;}
```

用于处理 WebSphere MQ 调用的适配器子任务数。

```
public int ChannelInitiatorDispatchers {get;}
```

要用于通道启动程序的分派器数。

```
public int ChannelInitiatorTraceAutoStart {get;}
```

指定通道启动程序跟踪是否自动启动。

```
public int ChannelInitiatorTraceTableSize {get;}
```

通道发起方的跟踪数据空间大小 (以兆字节为单位)。

```
public int ChannelMonitoring {get;}
```

是否使用通道监视。

```
public int ChannelStatistics {get;}
```

控制通道的统计数据的收集。

```
public int CharacterSet {get;}
```

返回队列管理器的编码字符集标识 (CCSID)。CharacterSet 由队列管理器用于应用程序编程接口中的所有字符串字段。

```
public int ClusterSenderMonitoring {get;}
```

控制自动定义的集群发送方通道的联机监视数据收集。

```
public int ClusterSenderStatistics {get;}
```

控制自动定义的集群发送方通道的统计数据收集。

```
public int ClusterWorkLoadMRU {get;}
```

出站集群通道的最大数量。

```
public int ClusterWorkLoadUseQ {get;}
```

MQQueue 属性 ClusterWorkLoadUseQ 的缺省值 (如果它指定的值为 QMGR)。

public int CommandEvent {get;}

指定是否生成命令事件。

public string CommandInputQueueName {get;}

返回队列管理器上定义的命令输入队列的名称。应用程序可以将命令发送到此队列 (如果已授权)。

public int CommandLevel {get;}

指示队列管理器的功能级别。对应于特定功能级别的功能集取决于平台。在特定平台上,您可以依赖于每个队列管理器,这些队列管理器支持所有队列管理器公共的最低功能级别的功能。

public int CommandLevel {get;}

命令服务器是否在队列管理器启动时自动启动。

public string DNSGroup {get;}

处理队列共享组的进站传输的 TCP 侦听器必须连接的组的名称。在使用工作负载管理器 for Dynamic Domain Name Services Support (DDNS) 时,它将加入此组。

public int DNSWLM {get;}

处理队列共享组的进站传输的 TCP 侦听器是否必须向工作负载管理器 for DDNS 注册。

public int IPAddressVersion {get;}

要用于通道连接的 IP 协议 (IPv4 或 IPv6)。

public boolean IsConnected {get;}

返回 isConnected 的值。

如果为 true,那么表示已建立与队列管理器的连接,并且不知道该连接已断开。对 IsConnected 的任何调用都不会主动尝试访问队列管理器,因此物理连接可能会中断,但 IsConnected 仍会返回 true。仅在队列管理器上执行活动 (例如,放置消息,获取消息) 时,才会更新 IsConnected 状态。

如果为 false,那么表示尚未与队列管理器建立连接,或者已断开连接,或者已断开连接。

public int KeepAlive {get;}

指定是否使用 TCP KEEPALIVE 工具来检查连接的另一端是否仍然可用。如果该通道不可用,那么将关闭该通道。

public int ListenerTimer {get;}

在 APPC 或 TCP/IP 故障后,WebSphere MQ 尝试重新启动侦听器的时间间隔 (以秒为单位)。

public int LoggerEvent {get;}

是否生成记录器事件。

public string LU62ARMSuffix {get;}

SYS1.PARMLIB。此后缀命名此通道启动程序的 LUADD。当自动重新启动管理器 (ARM) 重新启动通道启动程序时,将发出 z/OS 命令 SET APPC=xx。

public string LUGroupName {get; z/os}

要用于处理队列共享组的进站传输的 LU 6.2 侦听器使用的通用 LU 名。

public string LUName {get;}

要用于出站 LU 6.2 传输的 LU 的名称。

public int MaximumActiveChannels {get;}

在任一时刻可以处于活动状态的最大通道数。

public int MaximumCurrentChannels {get;}

可以随时处于当前状态的最大通道数 (包括具有已连接客户机的服务器连接通道)。

public int MaximumLU62Channels {get;}

可以是当前或可连接的使用 LU 6.2 传输协议的客户机的最大通道数。

public int MaximumMessageLength {get;}

返回队列管理器可处理的消息的最大长度 (以字节计)。无法定义最大消息长度大于 MaximumMessageLength 的队列。

public int MaximumPriority {get;}

返回队列管理器支持的最大消息优先级。优先级范围从零 (最低) 到此值。如果在与队列管理器断开连接后调用此方法,那么将抛出 MQException。

public int MaximumTCPChannels {get;}

可以是使用 TCP/IP 传输协议的当前或可连接的客户机的最大通道数。

public int MQIAccounting {get;}

控制 MQI 数据的记帐信息的收集。

public int MQIStatistics {get;}

控制队列管理器统计监视信息的收集。

public int OutboundPortMax {get;}

绑定传出通道时要使用的端口号范围内的最大值。

public int OutboundPortMin {get;}

绑定传出通道时要使用的端口号范围内的最小值。

public int QueueAccounting {get;}

是否将类 3 记帐 (线程级和队列级记帐) 数据用于所有队列。

public int QueueMonitoring {get;}

控制队列联机监视数据的收集。

public int QueueStatistics {get;}

控制队列的统计数据收集。

public int ReceiveTimeout {get;}

TCP/IP 通道在返回到不活动状态之前等待从其伙伴接收数据 (包括脉动信号) 的时间长度。

public int ReceiveTimeoutMin {get;}

TCP/IP 通道在返回到不活动状态之前等待从其伙伴接收数据 (包括脉动信号) 的最小时间长度。

public int ReceiveTimeoutType {get;}

要应用于 ReceiveTimeout 中的值的限定符。

public int SharedQueueQueueManagerName {get;}

指定如何将消息传递到共享队列。如果 put 指定与目标队列管理器相同的队列共享组中的另一个队列管理器, 那么将通过两种方式传递消息:

MQC.MQSQQM_USE

在将消息放入共享队列之前, 将消息传递到对象队列管理器。

MQCMQSQQM_IGNORE

消息直接放在共享队列上。

public int SSLEvent {get;}

是否生成 SSL 事件。

public int SSLFips {get;}

如果在 WebSphere MQ (而不是加密硬件) 中执行密码术, 那么是否仅使用 FIPS 认证的算法。

public int SSLKeyResetCount {get;}

指示在重新协商密钥之前在 SSL 对话中发送和接收的未加密字节数。

public int ClusterSenderStatistics {get;}

指定连续收集统计信息之间的时间间隔 (以分钟计)。

public int SyncpointAvailability {get;}

指示队列管理器是否支持使用 MQQueue.get 和 MQQueue.put 方法的工作单元和同步点。

public string TCPName {get;}

要使用的唯一或缺省 TCP/IP 系统的名称, 具体取决于 TCPStackType 的值。

public int TCPStackType {get;}

指定通道启动程序是否仅使用 TCPName 中指定的 TCP/IP 地址空间。或者, 通道启动程序可以绑定到任何 TCP/IP 地址。

public int TraceRouteRecording {get;}

控制路由跟踪信息的记录。

方法

```
public MQProcess AccessProcess(string processName, int openOptions);  
public MQProcess AccessProcess(string processName, int openOptions, string  
queueManagerName, string alternateUserId);
```

抛出 MQException。

访问此队列管理器上的 WebSphere MQ 进程以查询进程属性。

processName

要打开的进程的名称。

openOptions

用于控制进程打开的选项。可以添加或使用按位 OR 组合的有效选项包括:

- MQC.MQOO_FAIL_IF QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName

在其中定义进程的队列管理器的名称。如果队列管理器与进程正在访问的队列管理器相同，那么可以保留空白或空的队列管理器名称。

alternateUserId

如果在 *openOptions* 参数中指定了 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 将指定用于检查操作授权的备用用户标识。如果未指定 MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 可以为空或为空。

如果未指定 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么将使用缺省用户权限来连接到队列管理器。

```
public MQQueue AccessQueue(string queueName, int openOptions);  
public MQQueue AccessQueue(string queueName, int openOptions, string  
queueManagerName, string dynamicQueueName, string alternateUserId);
```

抛出 MQException。

访问此队列管理器上的队列。

您可以获取或浏览消息，放置消息，查询队列的属性或设置队列的属性。如果指定的队列是模型队列，那么将创建动态本地队列。查询生成的 MQQueue 对象的 name 属性以找出动态队列的名称。

queueName

要打开的队列的名称。

openOptions

用于控制队列打开的选项。

MQC.MQOO_ALTERNATE_USER_AUTHORITY

使用指定的用户标识进行验证。

MQC.MQOO_BIND_AS_QDEF

对队列使用缺省绑定。

MQC.MQOO_BIND_NOT_FIXED

请勿绑定到特定目标。

MQC.MQOO_BIND_ON_OPEN

打开队列时将句柄绑定到目标。

MQC.MQOO_BROWSE

打开以浏览消息。

MQC.MQOO_FAIL_IF QUIESCING

如果队列管理器正在停顿，那么失败。

MQC.MQOO_INPUT_AS_Q_DEF

打开以使用队列定义的缺省值获取消息。

MQC.MQOO_INPUT_SHARED

打开以获取具有共享访问权的消息。

MQC.MQOO_INPUT_EXCLUSIVE

打开以获取具有独占访问权的消息。

MQC.MQOO_INQUIRE

打开以进行查询-如果要查询属性，那么此选项是必需的。

MQC.MQOO_OUTPUT

打开以放置消息。

MQC.MQOO_PASS_ALL_CONTEXT

允许传递所有上下文。

MQC.MQOO_PASS_IDENTITY_CONTEXT

允许传递身份上下文。

MQC.MQOO_SAVE_ALL_CONTEXT

检索消息时保存上下文。

MQC.MQOO_SET

打开以设置属性-如果要设置属性，那么此选项是必需的。

MQC.MQOO_SET_ALL_CONTEXT

允许设置所有上下文。

MQC.MQOO_SET_IDENTITY_CONTEXT

允许设置身份上下文。

queueManagerName

定义队列的队列管理器的名称。完全为空白或空的名称表示 MQQueueManager 对象所连接的队列管理器。

dynamicQueueName

除非 *queueName* 指定模型队列的名称，否则将忽略 *dynamicQueueName*。如果这样做，那么 *dynamicQueueName* 将指定要创建的动态队列的名称。如果 *queueName* 指定模型队列的名称，那么空白或空名称无效。如果名称中的最后一个非空白字符是星号 *，那么队列管理器会将星号替换为字符串。这些字符保证为队列生成的名称在此队列管理器上是唯一的。

alternateUserId

如果在 *openOptions* 参数中指定了 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 指定用于检查打开权限的备用用户标识。如果未指定 MQC.MQOO_ALTERNATE_USER_AUTHORITY，那么 *alternateUserId* 可以留空或为空。


```

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);

```

访问此队列管理器上的主题。

MQTopic 对象与有时称为主题对象的管理主题对象密切相关。在输入时，topicObject 指向管理主题对象。MQTopic 构造函数从主题对象获取主题字符串，并将其与 topicName 组合以创建主题名称。topicObject 或 topicName 可以为空。主题名称与主题树匹配，并且在 topicObject 中返回最匹配的管理主题对象的名称。

与 MQTopic 对象关联的主题是组合两个主题字符串的结果。第一个主题字符串由 topicObject 标识的管理主题对象定义。第二个主题字符串为 topicString。生成的与 MQTopic 对象关联的主题字符串可以通过包含通配符来标识多个主题。

根据是打开主题以进行发布还是预订，您可以使用 MQTopic.Put 方法来发布主题，或者使用 MQTopic.Get 方法来接收有关主题的发布。如果要发布和预订同一主题，那么必须访问该主题两次，一次用于发布，一次用于预订。

如果为预订创建 MQTopic 对象，而不提供 MQDestination 对象，那么将采用受管预订。如果将队列作为 MQDestination 对象传递，那么将采用非受管预订。您必须确保设置的预订选项与受管或非受管的预订一致。

destination

destination 是 MQQueue 实例。通过提供 destination，MQTopic 将作为非受管预订打开。有关该主题的出版物将传递到作为 destination 访问的队列。

topicName

作为主题名称的第二部分的主题字符串。topicName 与 topicObject 管理主题对象中定义的主题字符串并置。您可以将 topicName 设置为 null，在这种情况下，主题名称由 topicObject 中的主题字符串定义。

topicObject

在输入时，topicObject 是包含构成主题名称第一部分的主题字符串的主题对象的名称。topicObject 中的主题字符串与 topicName 并置。用于构造主题名称的规则在 [组合主题字符串](#) 中定义。

在输出时，topicObject 包含在主题树中与主题名称所标识的主题最匹配的管理主题对象的名称。

openAs

访问要发布或预订的主题。该参数只能包含下列其中一个选项:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

组合用于控制发布或预订的主题打开的选项。使用 MQC.MQSO_* 常量来访问预订的主题，使用 MQC.MQOO_* 常量来访问发布的主题。

如果需要多个选项，请将值一起添加，或者使用按位 OR 运算符组合选项值。

alternateUserId

指定用于检查完成操作所需的权限的备用用户标识。如果在 options 参数中设置了 MQC.MQOO_ALTERNATE_USER_AUTHORITY 或 MQC.MQSO_ALTERNATE_USER_AUTHORITY，那么必须指定 *alternateUserId*。

subscriptionName

如果提供了选项 MQC.MQSO_DURABLE 或 MQC.MQSO_ALTER，那么需要 *subscriptionName*。在这两种情况下，都会隐式打开 MQTopic 以进行预订。如果设置了 MQC.MQSO_DURABLE，并且预订存在，或者设置了 MQC.MQSO_ALTER 并且预订不存在，那么将抛出异常。

properties

设置使用散列表列出的任何特殊预订属性。将使用输出值更新散列表中的指定条目。不会将条目添加到散列表以报告输出值。

- 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();

抛出 MQException

返回 MQAsyncStatus 对象，该对象表示队列管理器连接的异步活动。

public void Backout();

抛出 MQException。

回退自上次同步点以来在同步点内读取或写入的任何消息。

将从队列中除去使用 MQC.MQPMO_SYNCPOINT 标志集写入的消息。使用 MQC.MQGMO_SYNCPOINT 标志读取的消息将在它们来自的队列上恢复。如果消息是持久的，那么将记录更改。

对于可重新连接的客户机，MQRC_NONE 原因码将在重新连接成功后返回到客户机。

public void Begin();

抛出 MQException。

仅在服务器绑定方式下支持 Begin。它启动全局工作单元。

public void Commit();

抛出 MQException。

落实自上次同步点以来在同步点内读取或写入的任何消息。

使用 MQC.MQPMO_SYNCPOINT 标志集编写的消息可供其他应用程序使用。将删除使用 MQC.MQGMO_SYNCPOINT 标志集检索的消息。如果消息是持久的，那么将记录更改。

以下原因码将返回到可重新连接的客户机：

- MQRC_CALL_INTERRUPTED (如果在执行落实调用时连接丢失)。
- MQRC_BACKED_OUT (如果在重新连接后发出落实调用)。

Disconnect();

抛出 MQException。

关闭与队列管理器的连接。此队列管理器上访问的所有对象都不再可供此应用程序访问。要重新访问这些对象，请创建 MQQueueManager 对象。

通常，将落实作为工作单元一部分执行的任何工作。但是，如果工作单元由 .NET 管理，那么可能会回滚该工作单元。

```
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);
```

抛出 MQException。

将单个消息放在队列或主题上，而不首先创建 MQQueue 或 MQTopic 对象。

queueName

要将消息放入的队列的名称。

destinationName

目标对象的名称。它是队列或主题，具体取决于 *type* 的值。

type

目标对象的类型。不得组合选项。

MQC.MQOT_Q

队列

MQC.MQOT_TOPIC

Topic

queueManagerName

在其中定义队列的队列管理器或队列管理器别名的名称。如果指定了类型 MQC.MQOT_TOPIC，那么将忽略此参数。

如果队列是模型队列，并且解析的队列管理器名称不是此队列管理器，那么将抛出 MQException。

topicString

topicString 与 *destinationName* 主题对象中的主题名称组合。

如果 *destinationName* 是队列，那么将忽略 *topicString*。

message

要发送的消息。消息是输入/输出对象。

以下原因码将返回到可重新连接的客户机：

- MQRC_CALL_INTERRUPTED (如果对持久消息执行 Put 调用时连接中断)。
- MQRC_NONE 如果在对非持久消息执行 Put 调用时连接成功 (请参阅 [应用程序恢复](#))。

putMessageOptions

用于控制 put 操作的选项。

如果省略 `putMessageOptions`，那么将创建 `putMessageOptions` 的缺省实例。
`putMessageOptions` 是输入/输出对象。

如果在可重新连接的客户机中使用 `MQPMO_LOGICAL_ORDER` 选项，那么将返回 `MQRC_RECONNECT_INCOMPATIBLE` 原因码。

alternateUserId

指定在将消息放入队列时用于检查授权的备用用户标识。

如果未在 `putMessageOptions` 中设置 `MQC.MQOO_ALTERNATE_USER_AUTHORITY`，那么可以省略 `alternateUserId`。如果设置 `MQC.MQOO_ALTERNATE_USER_AUTHORITY`，那么还必须设置 `alternateUserId`。除非您还设置了 `MQC.MQOO_ALTERNATE_USER_AUTHORITY`，否则 `alternateUserId` 不会生效。

构造函数

```
public MQQueueManager();  
public MQQueueManager(string queueManagerName);  
public MQQueueManager(string queueManagerName, Int options);  
public MQQueueManager(string queueManagerName, Int options, string channel,  
string connName);  
public MQQueueManager(string queueManagerName, string channel, string  
connName);  
public MQQueueManager(string queueManagerName, System.Collections.Hashtable  
properties);
```

抛出 `MQException`。

创建与队列管理器的连接。在创建客户机连接或服务器连接之间进行选择。

尝试连接到队列管理器时，必须对队列管理器具有查询 (`inq`) 权限。如果没有查询权限，那么连接尝试将失败。

如果满足下列其中一个条件，那么将创建客户机连接：

1. `channel` 或 `connName` 在构造函数中指定。
2. `HostName`、`Port` 或 `Channel` 在 `properties` 中指定。
3. 指定了 `MQEnvironment.HostName`、`MQEnvironment.Port` 或 `MQEnvironment.Channel`。

连接属性的值以显示的顺序为缺省值。构造函数中的 `channel` 和 `connName` 优先于构造函数中的属性值。构造函数属性值优先于 `MQEnvironment` 属性。

主机名、通道名称和端口在 `MQEnvironment` 类中定义。

queueManagerName

要连接到的队列管理器或队列管理器组的名称。

省略该参数，或将其保留为空，或将其留空以进行缺省队列管理器选择。服务器上的缺省队列管理器连接到服务器上的缺省队列管理器。客户机连接上的缺省队列管理器连接是到侦听器所连接到的队列管理器。

options

指定 `MQCNO` 连接选项。这些值必须适用于所建立的连接类型。例如，如果为客户机连接指定以下服务器连接属性，那么将抛出 `MQException`。

- `MQC.MQCNO_FASTPATH_BINDING`
- `MQC.MQCNO_STANDARD_BINDING`

properties

属性参数采用一系列覆盖 `MQEnvironment` 设置的属性的键/值对；请参阅示例 [第 1166 页的『覆盖 MQEnvironment 属性』](#)。可以覆盖以下属性：

- `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

服务器连接通道的名称

connName

连接名称，格式为 *HostName* (端口)。

您可以提供 *hostnames* 和 *ports* 的列表，作为使用 CONNECTION_NAME_PROPERTY 的构造函数 MQQueueManager(String queueManagerName, Hashtable properties) 的自变量。

例如：

```

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);

```

进行连接尝试时，将按顺序处理连接名称列表。如果尝试连接到第一个主机名和端口失败，那么将尝试连接到第二对属性。客户机重复此过程，直到成功建立连接或耗尽列表为止。如果列表已耗尽，那么将向客户机应用程序返回相应的原因码和完成代码。

如果没有为连接名称提供端口号，那么将使用缺省端口 (在 mqclient.ini 中配置)。

设置连接列表

在设置自动客户机重新连接选项时，可以使用以下方法来设置连接列表：

通过 MQSERVER 设置连接列表

您可以通过命令提示符设置连接列表。

在命令提示符中，设置

```
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)
```

如果在 MQSERVER 中设置了连接，请不要在应用程序中设置该连接。

如果在应用程序中设置连接列表，那么应用程序将覆盖 MQSERVER 环境变量中设置的任何内容。

通过应用程序设置连接列表

您可以通过指定主机名和端口属性在应用程序中设置连接列表。

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

通过 app.config 设置连接列表

App.config 是一个 XML 文件，您可以在其中指定 "键/值" 对。

在连接列表中指定

```
<app.Settings>
<add key="Connection1" value="Hostname1(Port1)"/>
<add key="Connection2" value="Hostname2(Port2)"/>
</app.Settings>
```

例如：

```
<app.Settings>
<add key="Connection1" value="fred.mq.com(2966)"/>
<add key="Connection2" value="alex.mq.com(6533)"/>
</app.Settings>
```

您可以直接更改 app.config 文件中的连接列表。

通过 MQEnvironment 设置连接列表

要通过 MQEnvironment 设置连接列表，请使用 *ConnectionName* 属性。

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),alex.mq.com(5211);
```

ConnectionName 属性将覆盖 MQEnvironment 中设置的主机名和端口属性。

创建客户机连接

以下示例显示如何创建与队列管理器的客户机连接。您可以通过在创建新的 MQQueueManager 对象之前设置 MQEnvironment 变量来创建客户机连接。

```
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");
```

图 43: 客户机连接

覆盖 MQEnvironment 属性

以下示例显示如何使用散列表中定义的用户标识和密码来创建队列管理器。

```

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);
}

```

图 44: 覆盖 *MQEnvironment* 属性

创建可重新连接的连接

以下示例显示了如何自动将客户机重新连接到队列管理器。

```

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);

```

图 45: 自动将客户机重新连接到队列管理器

MQSubscription.NET 类

使用 *MQSubscription* 来请求将保留发布发送给订户。*MQSubscription* 是针对预订打开的 *MQTopic* 对象的属性。

类

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQSubscription

```

```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```

- [第 1167 页的『属性』](#)
- [第 1167 页的『方法』](#)
- [第 1168 页的『构造函数』](#)

属性

使用 *MQManagedObject* 类访问预订属性; 请参阅 [第 1130 页的『属性』](#)。

方法

使用 *MQManagedObject* 类访问预订 *Inquire*, *Set* 和 *Get* 方法; 请参阅 [第 1131 页的『方法』](#)。

```
public int RequestPublicationUpdate(int options);
```

抛出 MQException。

请求当前主题的更新发布。如果队列管理器具有该主题的保留发布，那么会将这些发布发送给订户。

在调用 RequestPublicationUpdate 之前，请打开预订主题以获取 MQSubscription 对象。

通常，使用 MQC.MQSO_PUBLICATIONS_ON_REQUEST 选项打开预订。如果主题字符串中不存在通配符，那么将仅发送一个发布内容作为此调用的结果。如果主题字符串包含通配符，那么可能会发送许多发布内容。此方法返回发送到预订队列的保留发布数。无法保证接收到这许多发布，尤其是当它们是非持久消息时。

options

MQC.MQSRO_FAIL_IF QUIESCING

如果队列管理器处于停顿状态，那么此方法将失败。在 z/OS 上，对于 CICS 或 IMS 应用程序，如果连接处于停顿状态，那么 MQC.MQSRO_FAIL_IF QUIESCING 还会强制该方法失败。

MQC.MQSRO_NONE

未指定任何选项。

构造函数

无 Public 构造函数。

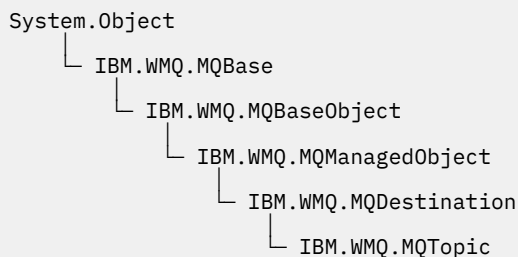
将在针对预订打开的 MQTopic 对象的 SubscriptionReference 属性中返回 MQSubscription 对象。

调用 RequestPublicationUpdate 方法。MQSubscription 是 MQManagedObject 的子类。使用该引用来访问 MQManagedObject 的属性和方法。

MQTopic .NET 类

使用 MQTopic 来发布或预订主题上的消息，或者查询或设置主题的属性。使用构造函数或 MQQueueManager.AccessTopic 方法创建用于发布或预订的 MQTopic 对象。

类



```
public class IBM.WMQ.MQTopic extends IBM.WMQ.MQDestination;
```

- [第 1168 页的『属性』](#)
- [第 1169 页的『方法』](#)
- [第 1170 页的『构造函数』](#)

属性

获取属性时抛出 MQException 的测试。

```
public Boolean IsDurable {get;}
```

只读属性，如果预订是持久的，那么返回 True，否则返回 False。如果打开主题以进行发布，那么将忽略该属性并始终返回 False。

public Boolean IsManaged {get};

如果预订由队列管理器管理，那么返回 True 的只读属性，否则返回 False。如果打开该主题以进行发布，那么将忽略该属性，并且该属性将始终返回 False。

public Boolean IsSubscribed {get};

只读属性，如果打开主题以进行预订，那么返回 True；如果打开主题以进行发布，那么返回 False。

public MQSubscription SubscriptionReference {get};

只读属性，返回与为预订打开的主题对象关联的 MQSubscription 对象。如果要修改关闭选项或启动任何对象方法，那么该引用可用。

public MQDestination UnmanagedDestinationReference {get};

用于返回与非受管预订关联的 MQQueue 的只读属性。它是创建主题对象时指定的目标。对于为发布或使用受管预订打开的任何主题对象，此属性返回 null。

方法

public void Put(MQMessage message);

public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);

抛出 MQException。

将消息发布到主题。

完成 Put 调用后对 MQMessage 对象的修改不会影响 WebSphere MQ 队列或发布主题上的实际消息。

Put 会更新 MQMessage 对象的 MessageId 和 CorrelationId 属性，并且不会清除消息数据。进一步的 Put 或 Get 调用引用 MQMessage 对象中的更新信息。例如，在以下代码片段中，第一条消息包含 a 和第二条 ab。

```
msg.WriteString("a");
q.Put(msg, pmo);
msg.WriteString("b");
q.Put(msg, pmo);
```

message

包含消息描述符数据和要发送的消息的 MQMessage 对象。由于此方法，可以更改消息描述符。在此方法完成后，消息描述符中的值是已放入队列或已发布到主题的值。

以下原因码将返回到可重新连接的客户机：

- MQRC_CALL_INTERRUPTED：如果在对持久消息运行 Put 调用时连接中断，并且重新连接成功。
- MQRC_NONE (如果在对非持久消息运行 Put 调用时连接成功) (请参阅 [应用程序恢复](#))。

putMessageOptions

用于控制 put 操作的选项。

如果未指定 *putMessageOptions*，那么将使用 MQPutMessageOptions 的缺省实例。

如果在可重新连接的客户机中使用 MQPMO_LOGICAL_ORDER 选项，那么将返回 MQRC_RECONNECT_INCOMPATIBLE 原因码。

注：为了实现简单性和性能，如果要将单个消息放入队列，请使用 MQQueueManager.Put 对象。您应该具有此对象的 MQQueue 对象。

public void Get(MQMessage message);

public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);

public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int MaxMsgSize);

抛出 MQException。

从主题检索消息。

此方法使用 MQGetMessageOptions 的缺省实例来执行 get 操作。使用的消息选项为 MQGMO_NOWAIT。

如果 get 失败，那么 MQMessage 对象保持不变。如果成功，那么 MQMessage 的消息描述符和消息数据部分将替换为来自入局消息的消息描述符和消息数据。

从特定 `MQQueueManager` 对 WebSphere MQ 的所有调用都是同步的。因此，如果执行获取并等待，那么将阻止使用相同 `MQQueueManager` 的所有其他线程进行进一步的 WebSphere MQ 调用，直到完成 `Get` 调用为止。如果需要多个线程来同时访问 WebSphere MQ，那么每个线程都必须创建自己的 `MQQueueManager` 对象。

message

包含消息描述符和返回的消息数据。消息描述符中的某些字段是输入参数。确保根据需要设置 `MessageId` 和 `CorrelationId` 输入参数非常重要。

对于在 `MQGM_SYNCPOINT` 下接收到的消息，可重新连接的客户机在成功重新连接后返回原因码 `MQRC_BACKED_OUT`。

getMessageOptions

控制获取操作的选项。

使用选项 `MQC.MQGM_CONVERT` 可能会导致在从单字节字符代码转换为双字节代码时发生异常，原因码为 `MQC.MQRC_CONVERTED_STRING_TOO_BIG`。在这种情况下，会将消息复制到缓冲区中而不进行转换。

如果未指定 `getMessageOptions`，那么使用的消息选项为 `MQGM_NOWAIT`。

如果在可重新连接的客户机中使用 `MQGM_LOGICAL_ORDER` 选项，那么将返回 `MQRC_RECONNECT_INCOMPATIBLE` 原因码。

MaxMsgSize

此消息对象要接收的最大消息。如果队列上的消息大于此大小，那么将发生以下两种情况之一：

- 如果在 `MQGetMessageOptions` 对象中设置了 `MQGM_ACCEPT_TRUNCATED_MSG` 标志，那么将使用尽可能多的消息数据来填充消息。抛出异常时带有 `MQCC_WARNING` 完成代码和 `MQRC_TRUNCATED_MSG_ACCEPTED` 原因码。
- 如果未设置 `MQGM_ACCEPT_TRUNCATED_MSG` 标志，那么消息将保留在队列中。抛出异常时带有 `MQCC_WARNING` 完成代码和 `MQRC_TRUNCATED_MSG_FAILED` 原因码。

如果未指定 `MaxMsgSize`，那么将检索整个消息。

构造函数

```
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);
```

访问 `queueManager` 上的主题。

`MQTopic` 对象与有时称为主题对象的管理主题对象密切相关。在输入时，`topicObject` 指向管理主题对象。`MQTopic` 构造函数从主题对象获取主题字符串，并将其与 `topicName` 组合以创建主题名称。`topicObject` 或 `topicName` 可以为空。主题名称与主题树匹配，并且在 `topicObject` 中返回最匹配的 `管理主题对象` 的名称。

与 MQTopic 对象关联的主题是组合两个主题字符串的结果。第一个主题字符串由 *topicObject* 标识的管理主题对象定义。第二个主题字符串为 *topicString*。生成的与 MQTopic 对象关联的主题字符串可以通过包含通配符来标识多个主题。

根据是打开主题以进行发布还是预订，您可以使用 MQTopic.Put 方法来发布主题，或者使用 MQTopic.Get 方法来接收有关主题的发布。如果要发布和预订同一主题，那么必须访问该主题两次，一次用于发布，一次用于预订。

如果为预订创建 MQTopic 对象，而不提供 MQDestination 对象，那么将采用受管预订。如果将队列作为 MQDestination 对象传递，那么将采用非受管预订。您必须确保设置的预订选项与受管或非受管的预订一致。

queueManager

要访问主题的队列管理器。

destination

destination 是 MQQueue 实例。通过提供 *destination*，MQTopic 将作为非受管预订打开。有关该主题的出版物将传递到作为 *destination* 访问的队列。

topicName

作为主题名称的第二部分的主题字符串。*topicName* 与 *topicObject* 管理主题对象中定义的主题字符串并置。您可以将 *topicName* 设置为 null，在这种情况下，主题名称由 *topicObject* 中的主题字符串定义。

topicObject

在输入时，*topicObject* 是包含构成主题名称第一部分的主题字符串的主题对象的名称。*topicObject* 中的主题字符串与 *topicName* 并置。用于构造主题名称的规则在 [组合主题字符串](#) 中定义。

在输出时，*topicObject* 包含在主题树中与主题名称所标识的主题最匹配的管理主题对象的名称。

openAs

访问要发布或预订的主题。该参数只能包含下列其中一个选项：

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

组合用于控制发布或预订的主题打开的选项。使用 MQC.MQSO_* 常量来访问预订的主题，使用 MQC.MQOO_* 常量来访问发布的主题。

如果需要多个选项，请将值一起添加，或者使用按位 OR 运算符组合选项值。

alternateUserId

指定用于检查完成操作所需的权限的备用用户标识。如果在 options 参数中设置了 MQC.MQOO_ALTERNATE_USER_AUTHORITY 或 MQC.MQSO_ALTERNATE_USER_AUTHORITY，那么必须指定 *alternateUserId*。

subscriptionName

如果提供了选项 MQC.MQSO_DURABLE 或 MQC.MQSO_ALTER，那么需要 *subscriptionName*。在这两种情况下，都会隐式打开 MQTopic 以进行预订。如果设置了 MQC.MQSO_DURABLE，并且预订存在，或者设置了 MQC.MQSO_ALTER 并且预订不存在，那么将抛出异常。

properties

设置使用散列表列出的任何特殊预订属性。将使用输出值更新散列表中的指定条目。不会将条目添加到散列表以报告输出值。

- 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);

```

访问此队列管理器上的主题。

MQTopic 对象与有时称为主题对象的管理主题对象密切相关。在输入时，topicObject 指向管理主题对象。MQTopic 构造函数从主题对象获取主题字符串，并将其与 topicName 组合以创建主题名称。topicObject 或 topicName 可以为空。主题名称与主题树匹配，并且在 topicObject 中返回最匹配的管理主题对象的名称。

与 MQTopic 对象关联的主题是组合两个主题字符串的结果。第一个主题字符串由 topicObject 标识的管理主题对象定义。第二个主题字符串为 topicString。生成的与 MQTopic 对象关联的主题字符串可以通过包含通配符来标识多个主题。

根据是打开主题以进行发布还是预订，您可以使用 MQTopic.Put 方法来发布主题，或者使用 MQTopic.Get 方法来接收有关主题的发布。如果要发布和预订同一主题，那么必须访问该主题两次，一次用于发布，一次用于预订。

如果为预订创建 MQTopic 对象，而不提供 MQDestination 对象，那么将采用受管预订。如果将队列作为 MQDestination 对象传递，那么将采用非受管预订。您必须确保设置的预订选项与受管或非受管的预订一致。

destination

destination 是 MQQueue 实例。通过提供 destination，MQTopic 将作为非受管预订打开。有关该主题的出版物将传递到作为 destination 访问的队列。

topicName

作为主题名称的第二部分的主题字符串。topicName 与 topicObject 管理主题对象中定义的主题字符串并置。您可以将 topicName 设置为 null，在这种情况下，主题名称由 topicObject 中的主题字符串定义。

topicObject

在输入时，topicObject 是包含构成主题名称第一部分的主题字符串的主题对象的名称。topicObject 中的主题字符串与 topicName 并置。用于构造主题名称的规则在 [组合主题字符串](#) 中定义。

在输出时，topicObject 包含在主题树中与主题名称所标识的主题最匹配的管理主题对象的名称。

openAs

访问要发布或预订的主题。该参数只能包含下列其中一个选项:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

组合用于控制发布或预订的主题打开的选项。使用 `MQC.MQSO_*` 常量来访问预订的主题，使用 `MQC.MQOO_*` 常量来访问发布的主题。

如果需要多个选项，请将值一起添加，或者使用按位 OR 运算符组合选项值。

alternateUserId

指定用于检查完成操作所需的权限的备用用户标识。如果在 `options` 参数中设置了 `MQC.MQOO_ALTERNATE_USER_AUTHORITY` 或 `MQC.MQSO_ALTERNATE_USER_AUTHORITY`，那么必须指定 `alternateUserId`。

subscriptionName

如果提供了选项 `MQC.MQSO_DURABLE` 或 `MQC.MQSO_ALTER`，那么需要 `subscriptionName`。在这两种情况下，都会隐式打开 `MQTopic` 以进行预订。如果设置了 `MQC.MQSO_DURABLE`，并且预订存在，或者设置了 `MQC.MQSO_ALTER` 并且预订不存在，那么将抛出异常。

properties

设置使用散列表列出的任何特殊预订属性。将使用输出值更新散列表中的指定条目。不会将条目添加到散列表以报告输出值。

- `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`

IMQObjectTrigger.NET 接口

实现 `IMQObjectTrigger` 以处理 `runmqdmn`.NET 监视器传递的消息。

接口

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

根据 `runmqdmn` 命令中是否指定了同步点控制，将在 `Execute` 方法返回之前或之后从队列中除去消息。

方法

```
void Execute (MQQueueManager queueManager, MQQueue queue, MQMessage message, string param);
```

queueManager

托管受监视队列的队列管理器。

queue

正在监视的队列。

message

从队列读取的消息。

param

从 `UserParameter` 传递的数据。

MQC.NET 接口

通过在常量名称前面添加 `MQC.` 来引用 `MQI` 常量。 `MQC` 定义 `MQI` 使用的所有常量。

接口

```
System.Object
└─ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

示例

```
MQQueue queue;
queue.closeOptions = MQC.MQCO_DELETE;
```

.NET 应用程序的字符集标识

可以选择对 .NET IBM WebSphere MQ 消息进行编码的字符集的描述

| 字符集 | 描述 |
|------|------------------------------|
| 37 | ibm037 |
| 437 | ibm437 /PC 原始 |
| 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 希腊语 |
| 775 | ibm775 /PC 波罗的语 |
| 813 | iso-8859-7 /greek/ ibm813 |
| 838 | ibm838 |
| 850 | ibm850 /PC 拉丁语 1 |
| 852 | ibm852 /PC 拉丁语 2 |
| 855 | ibm855 /PC 西里尔语 |
| 856 | ibm856 |
| 857 | ibm857 /PC 土耳其语 |
| 860 | ibm860 /PC 葡萄牙语 |

| 字符集 | 描述 |
|-----|------------------------------|
| 861 | ibm861 /PC 冰岛语 |
| 862 | ibm862 /PC 希伯来语 |
| 863 | ibm863 /PC 加拿大法语 |
| 864 | ibm864 /PC 阿拉伯语 |
| 865 | ibm865 /PC 北欧版 |
| 866 | ibm866 /PC 俄语 |
| 868 | ibm868 |
| 869 | ibm869 /PC 现代希腊语 |
| 870 | ibm870 |
| 871 | ibm871 |
| 874 | ibm874 |
| 875 | ibm875 |
| 912 | iso-8859-2 / latin2 / ibm912 |
| 913 | iso-8859-3 / latin3 / ibm913 |
| 914 | iso-8859-4 / latin4 / ibm914 |
| 915 | iso-8859-5 /西里尔文/ ibm915 |
| 916 | iso-8859-8 /希伯来语/ ibm916 |
| 918 | ibm918 |
| 920 | iso-8859-9 / latin5 / ibm920 |
| 921 | ibm921 |
| 922 | ibm922 |
| 930 | ibm930 |
| 932 | PC 日语 |
| 933 | ibm933 |
| 935 | ibm935 |
| 937 | ibm937 |
| 939 | ibm939 |
| 942 | ibm942 |
| 943 | ibm943 |
| 948 | ibm948 |
| 949 | ibm949 |
| 950 | ibm950 /Big 5 繁体中文 |
| 954 | eucjis |
| 964 | ibm964 /CNS 11643 繁体中文 |
| 970 | ibm970 |

| 字符集 | 描述 |
|-------|-----------------------------|
| 1006 | ibm1006 |
| 1025 | ibm1025 |
| 1026 | ibm1026 |
| 1089 | iso-8859-6 /arabic/ ibm1089 |
| 1097 | ibm1097 |
| 1098 | ibm1098 |
| 1112 | ibm1112 |
| 1122 | ibm1122 |
| 1123 | ibm1123 |
| 1124 | ibm1124 |
| 1250 | Windows 拉丁语 2 |
| 1251 | Windows 西里尔字母 |
| 1252 | Windows 拉丁语 1 |
| 1253 | Windows 希腊语 |
| 1254 | Windows 土耳其语 |
| 1255 | Windows 希伯来语 |
| 1256 | Windows 阿拉伯语 |
| 1257 | Windows 波罗的语 |
| 1258 | Windows 越南语 |
| 1381 | ibm1381 |
| 1383 | ibm1383 |
| 2022 | JIS |
| 5601 | ksc-5601 韩国语 |
| 33722 | ibm33722 |

IBM WebSphere MQ C++ 类

IBM WebSphere MQ C++ 类封装了 IBM WebSphere MQ 消息队列接口 (MQI)。有一个单独的 C++ 头文件 **imqi.hpp**，它涵盖所有这些类。

对于每个类，将显示以下信息：

类层次结构图

类图，显示类与其直接父类 (如果有) 的继承关系中的类。

其他相关类

文档链接到其他相关类 (例如，父类) 以及方法特征符中使用的对象类。

对象属性

类的属性。这些属性是对为任何父类定义的属性的补充。许多属性反映 WebSphere MQ 数据结构成员 (请参阅第 1178 页的『C++ 和 MQI 交叉引用』)。有关详细描述，请参阅第 701 页的『对象的属性』。

构造函数

用于创建类对象的特殊方法的特征符。

对象方法(公用)

需要类实例进行操作且没有使用限制的方法的特征符。

如果适用, 那么还会显示以下信息:

类方法(public)

方法的特征符, 这些方法不需要类的实例进行操作, 并且没有使用限制。

重载(父类)方法

在父类中定义的那些虚拟方法的特征符, 但对于此类表现出不同的多态行为。

对象方法(protected)

需要类实例进行操作的方法的特征符, 保留供派生类的实现使用。此部分仅对类写程序感兴趣, 而与类用户不同。

对象数据(受保护)

可用于派生类实现的对象实例数据的实现详细信息。此部分仅对类写程序感兴趣, 而与类用户不同。

原因码

MQRC_* 值(请参阅 API 原因码), 可以从失败的方法中获取这些值。要获取针对类的对象可能发生的原因码的详尽列表, 请参阅父类文档。记录的类原因码列表不包含父类的原因码。

注:

1. 这些类的对象不是线程安全的。这可确保最佳性能, 但请注意不要从多个线程访问任何对象。
2. 对于多线程程序, 建议对每个线程使用单独的 ImqQueueManager 对象。每个管理器对象都必须有自己的独立其他对象集合, 确保不同线程中的对象相互隔离。

这些类是:

- [第 1191 页的『ImqAuthentication 记录 C++ 类』](#)
- [第 1193 页的『ImqBinary C++ 类』](#)
- [第 1195 页的『ImqCache C++ 类』](#)
- [第 1197 页的『ImqChannel C++ 类』](#)
- [第 1202 页的『ImqCICSBridge 头 C++ 类』](#)
- [第 1208 页的『ImqDeadLetterHeader C++ 类』](#)
- [第 1211 页的『ImqDistribution 列出 C++ 类』](#)
- [第 1212 页的『ImqError C++ 类』](#)
- [第 1213 页的『ImqGetMessageOptions C++ 类』](#)
- [第 1216 页的『ImqHeader C++ 类』](#)
- [第 1218 页的『ImqIMSBridge 头 C++ 类』](#)
- [第 1220 页的『ImqItem C++ 类』](#)
- [第 1222 页的『ImqMessage C++ 类』](#)
- [第 1228 页的『ImqMessageTracker C++ 类』](#)
- [第 1231 页的『ImqNamelist C++ 类』](#)
- [第 1233 页的『ImqObject C++ 类』](#)
- [第 1238 页的『ImqProcess C++ 类』](#)
- [第 1239 页的『ImqPutMessageOptions C++ 类』](#)
- [第 1241 页的『ImqQueue C++ 类』](#)
- [第 1251 页的『ImqQueueManager C++ 类』](#)
- [第 1266 页的『ImqReference 头 C++ 类』](#)
- [第 1269 页的『ImqString C++ 类』](#)
- [第 1273 页的『ImqTrigger C++ 类』](#)
- [第 1276 页的『ImqWork 头 C++ 类』](#)

C++ 和 MQI 交叉引用

此主题集合包含与 MQI 相关的 C++ 信息。

请与第 214 页的『MQI 中使用的数据类型』一起阅读本信息。

此表将 MQI 数据结构与 C++ 类和包含文件相关。以下主题显示每个 C++ 类的交叉引用信息。这些交叉引用与使用底层 WebSphere MQ 过程接口相关。类 ImqBinary, ImqDistributionList 和 ImqString 没有属于此类别的属性，将排除这些属性。

| 数据结构 | 类 | 包含文件 |
|---------------------|----------------------|------------|
| MQAIR | ImqAuthentication 记录 | imqair.hpp |
| | ImqBinary | imqbin.hpp |
| | ImqCache | imqcac.hpp |
| MQCD | ImqChannel | imqchl.hpp |
| MQCIH | ImqCICSBridge 头 | imqcih.hpp |
| MQDLH | ImqDeadLetterHeader | imqdlh.hpp |
| MQOR | ImqDistribution 列表 | imqdst.hpp |
| | ImqError | imqerr.hpp |
| MQGMO | ImqGetMessageOptions | imqgmo.hpp |
| | ImqHeader | imqhdr.hpp |
| MQIIH | ImqIMSBridge 头 | imqiih.hpp |
| | ImqItem | imqitm.hpp |
| MQMD | ImqMessage | imqmsg.hpp |
| | ImqMessage 跟踪程序 | imqmtr.hpp |
| | ImqNamelist | imqnml.hpp |
| MQOD 和 MQRR | ImqObject | imqobj.hpp |
| MQPMO, MQPMR 和 MQRR | ImqPutMessageOptions | imqpmo.hpp |
| | ImqProcess | imqpro.hpp |
| | ImqQueue | imqque.hpp |
| MQBO, MQCNO 和 MQCSP | ImqQueue 管理器 | imqmgr.hpp |
| MQRMH | ImqReference 头 | imqrfh.hpp |
| | ImqString | imqstr.hpp |
| MQTM | ImqTrigger | imqtrg.hpp |
| MQTMC | | |
| MQTMC2 | ImqTrigger | imqtrg.hpp |
| MQXQH | | |
| MQWIH | ImqWork 头 | imqwih.hpp |

ImqAuthentication 记录交叉引用

ImqAuthenticationRecord C++ 类的属性，数据结构，字段和调用的交叉引用。

| 属性 | 数据结构 | 字段 | 电话 |
|------|-------|------------------|--------|
| 连接名称 | MQAIR | AuthInfoConnName | MQCONN |
| 密码 | MQAIR | LDAPPassword | MQCONN |
| 类型 | MQAIR | AuthInfoType | MQCONN |
| 用户名 | MQAIR | LDAPUserNamePtr | MQCONN |
| | MQAIR | LDAPUserName 偏移量 | MQCONN |
| | MQAIR | LDAPUserName 长度 | MQCONN |

ImqCache 交叉引用

ImqCache C++ 类的属性和调用的交叉引用。

| 属性 | 电话 |
|-------|---------------|
| 自动缓冲区 | MQGET |
| 缓冲区长度 | MQGET |
| 缓冲区指针 | MQGET 和 MQPUT |
| 数据长度 | MQGET |
| 数据偏移量 | MQGET |
| 数据指针 | MQGET |
| 消息长度 | MQGET 和 MQPUT |

ImqChannel 交叉引用

ImqChannel C++ 类的属性，数据结构，字段和调用的交叉引用。

| 属性 | 数据结构 | 字段 | 电话 |
|----------|------|--------------------|--------|
| 批处理脉动信号 | MQCD | BatchHeartbeat | MQCONN |
| 通道名称 | MQCD | ChannelName | MQCONN |
| 连接名称 | MQCD | ConnectionName | MQCONN |
| | MQCD | ShortConnection 名称 | MQCONN |
| 头压缩 | MQCD | HdrComp 列表 | MQCONN |
| 心跳间隔 | MQCD | HeartbeatInterval | MQCONN |
| 保持活动时间间隔 | MQCD | KeepAliveInterval | MQCONN |
| 本地地址 | MQCD | LocalAddress | MQCONN |
| 最大消息长度 | MQCD | MaxMsgLength | MQCONN |
| 消息压缩 | MQCD | MsgComp 列表 | MQCONN |
| 方式名 | MQCD | ModeName | MQCONN |
| 密码 | MQCD | 密码 | MQCONN |
| 接收出口计数 | MQCD | | MQCONN |
| 接收出口名称 | MQCD | ReceiveExit | MQCONN |
| | MQCD | ReceiveExits 已定义 | MQCONN |

| 属性 | 数据结构 | 字段 | 电话 |
|----------------|------|--------------------|--------|
| | MQCD | ReceiveExitPtr | MQCONN |
| 接收用户数据 | MQCD | ReceiveUserData | MQCONN |
| | MQCD | ReceiveUserDataPtr | MQCONN |
| 安全出口名 | MQCD | SecurityExit | MQCONN |
| 安全性用户数据 | MQCD | SecurityUser 数据 | MQCONN |
| 发送出口计数 | MQCD | | MQCONN |
| 发送出口名称 | MQCD | SendExit | MQCONN |
| | MQCD | SendExits 已定义 | MQCONN |
| | MQCD | SendExitPtr | MQCONN |
| 发送用户数据 | MQCD | SendUserData | MQCONN |
| | MQCD | SendUserDataPtr | MQCONN |
| SSL CipherSpec | MQCD | sslCipher 规范 | MQCONN |
| SSL 客户机认证类型 | MQCD | sslClient 认证 | MQCONN |
| SSL 对等名称 | MQCD | sslPeerName | MQCONN |
| 事务程序名 | MQCD | TpName | MQCONN |
| 传输类型 | MQCD | TransportType | MQCONN |
| 用户标识 | MQCD | UserIdentifier | MQCONN |

ImqCICSBridge 头交叉引用

ImqCICSBridge 头 C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|---------------------|-------|---------------------|
| 网桥异常终止代码 | MQCIH | AbendCode |
| ADS 描述符 | MQCIH | AdsDescriptor |
| 注意标识 | MQCIH | AttentionId |
| 鉴别符 (authenticator) | MQCIH | Authenticator |
| 网桥完成代码 | MQCIH | BridgeCompletion 代码 |
| 网桥错误偏移量 | MQCIH | ErrorOffset |
| 网桥原因码 | MQCIH | BridgeReason |
| 网桥取消代码 | MQCIH | CancelCode |
| 会话式任务 | MQCIH | ConversationalTask |
| 光标位置 | MQCIH | CursorPosition |
| 设施令牌 | MQCIH | 设施 |
| 设施保留时间 | MQCIH | FacilityKeepTime |
| 设施, 例如 | MQCIH | FacilityLike |
| 函数 (function) | MQCIH | 函数 |
| 获取等待时间间隔 | MQCIH | GetWaitInterval |

| 属性 | 数据结构 | 字段 |
|---------|-------|-------------------|
| 链接类型 | MQCIH | LinkType |
| 下一个事务标识 | MQCIH | NextTransactionId |
| 输出数据长度 | MQCIH | OutputDataLength |
| 应答格式 | MQCIH | ReplyToFormat |
| 网桥返回码 | MQCIH | ReturnCode |
| 启动代码 | MQCIH | StartCode |
| 任务结束状态 | MQCIH | TaskEndStatus |
| 事务标识 | MQCIH | TransactionId |
| uow 控件 | MQCIH | UowControl |
| version | MQCIH | 版本 |

ImqDeadLetterHeader 交叉引用

ImqDeadLetterHeader C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|------------|-------|--------------|
| 死信原因码 | MQDLH | 原因 |
| 目标队列管理器名称 | MQDLH | DestQMgrName |
| 目标队列名称 | MQDLH | DestQName |
| PUT 应用程序名称 | MQDLH | PutApplName |
| PUT 应用程序类型 | MQDLH | PutApplType |
| PUT 日期 | MQDLH | PutDate |
| PUT 时间 | MQDLH | PutTime |

ImqError 交叉引用

ImqError C++ 类的属性和调用的交叉引用。

| 属性 | 电话 |
|------------------------|---|
| 完成代码 (completion code) | MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET |
| 原因码 (reason code) | MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET |

ImqGetMessageOptions 交叉引用

ImqGetMessageOptions C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|----------------------|-------|--------------|
| 集团状态 | MQGMO | GroupStatus |
| 匹配选项 | MQGMO | MatchOptions |
| 消息标记 (message token) | MQGMO | MessageToken |

| 属性 | 数据结构 | 字段 |
|---------|-------|----------------|
| 选项 | MQGMO | 选项 |
| 已解析队列名称 | MQGMO | ResolvedQName |
| 返回的长度 | MQGMO | ReturnedLength |
| 分段 | MQGMO | 分段 |
| 段状态 | MQGMO | SegmentStatus |
| | MQGMO | Signal1 |
| | MQGMO | Signal2 |
| 同步点参与 | MQGMO | 选项 |
| 等待时间间隔 | MQGMO | WaitInterval |

ImqHeader 交叉引用

ImqHeader C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|-----|---------------|----------------|
| 字符集 | MQDLH 和 MQIIH | CodedCharSetId |
| 编码 | MQDLH 和 MQIIH | 编码 |
| 格式 | MQDLH 和 MQIIH | 格式 |
| 头标志 | MQIIH 和 MQRMH | 标志 |

ImqIMSBridge 头交叉引用

ImqAuthenticationRecord C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|---------------------|-------|----------------|
| 鉴别符 (authenticator) | MQIIH | Authenticator |
| 落实方式 | MQIIH | CommitMode |
| 逻辑终端覆盖 | MQIIH | LTermOverride |
| 消息格式化服务映射名称 | MQIIH | MFSMapName |
| 应答格式 | MQIIH | ReplyToFormat |
| 安全范围 | MQIIH | SecurityScope |
| 事务实例标识 | MQIIH | TranInstanceId |
| 事务状态 | MQIIH | TranState |

ImqItem 交叉引用

ImqItem C++ 类的属性和调用的交叉引用。

| 属性 | 电话 |
|------|-------|
| 结构标识 | MQGET |

ImqMessage 交叉引用

ImqMessage C++ 类的属性，数据结构，字段和调用的交叉引用。

| 属性 | 数据结构 | 字段 | 电话 |
|----------------------|------|------------------|-------|
| 应用程序标识数据 | MQMD | ApplIdentityData | |
| 应用程序源数据 | MQMD | ApplOriginData | |
| 回退计数 | MQMD | BackoutCount | |
| 字符集 | MQMD | CodedCharSetId | |
| 编码 | MQMD | 编码 | |
| 到期 | MQMD | 到期 | |
| 格式 | MQMD | 格式 | |
| 消息标志 | MQMD | MsgFlags | |
| 消息类型 | MQMD | MsgType | |
| offset | MQMD | 偏移量 | |
| 原始长度 | MQMD | OriginalLength | |
| 持久性 | MQMD | 持久 | |
| priority | MQMD | 优先级 | |
| PUT 应用程序名称 | MQMD | PutApplName | |
| PUT 应用程序类型 | MQMD | PutApplType | |
| PUT 日期 | MQMD | PutDate | |
| PUT 时间 | MQMD | PutTime | |
| 应答队列管理器名称 | MQMD | ReplyToQMgr | |
| 应答队列名称 | MQMD | ReplyToQ | |
| 报告 | MQMD | 报告 | |
| 序号 (sequence number) | MQMD | MsgSeqNumber | |
| 消息总长度 | | DataLength | MQGET |
| 用户标识 | MQMD | UserIdentifier | |

ImqMessage 跟踪程序交叉引用

ImqMessageTracker C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|------|------|-----------------|
| 记帐标记 | MQMD | AccountingToken |
| 相关标识 | MQMD | CorrelId |
| 反馈 | MQMD | Feedback |
| 组标识 | MQMD | GroupId |
| 消息标识 | MQMD | MsgId |

ImqNamelist 交叉引用

ImqNamelist C++ 类的属性，查询和调用的交叉引用。

| 属性 | 查询 | 电话 |
|--------|--------------------|-------|
| 名称计数 | MQIA_NAME_COUNT | MQINQ |
| 名称列表名称 | MQCA_NAMELIST_NAME | MQINQ |

ImqObject 交叉引用

ImqObject C++ 类的属性，数据结构，字段，查询和调用的交叉引用。

| 属性 | 数据结构 | 字段 | 查询 | 电话 |
|-------------|---------|-----------------|---|---------------------|
| 变更日期 | | | MQCA_ALTERATION_DATE | MQINQ |
| 变更时间 | | | MQCA_ALTERATION_TIME | MQINQ |
| 备用用户标识 | MQOD | AlternateUserId | | |
| 备用安全标识 | | | | |
| 关闭选项 | | | | MQCLOSE |
| description | | | MQCA_Q_DESC , MQCA_Q_MGR_DESC 和 MQCA_PROCESS_DESC | MQINQ |
| 名 | MQOD | ObjectName | MQCA_Q_MGR_NAME , MQCQ_Q_NAME 和 MQCA_PROCESS_NAME | MQINQ |
| 打开选项 | | | | MQOPEN |
| 打开状态 | | | | MQOPEN 和 MQCLOSE |
| 队列管理器标识 | 队列管理器标识 | | MQCA_Q_MGR_IDENTIFIER | MQINQ |

ImqProcess 交叉引用

ImqAuthenticationRecord C++ 类的属性，查询和调用的交叉引用。

| 属性 | 查询 | 电话 |
|--------|----------------|-------|
| 应用程序标识 | MQCA_APPL_ID | MQINQ |
| 应用程序类型 | MQIA_APPL_TYPE | MQINQ |
| 环境数据 | MQCA_ENV_DATA | MQINQ |
| 用户数据 | MQCA_USER_DATA | MQINQ |

ImqPutMessageOptions 交叉引用

ImqAuthenticationRecord C++ 类的属性，数据结构和字段的交叉引用。

| 表 611: ImqPutMessageOptions 交叉引用 | | |
|----------------------------------|-------|------------------|
| 属性 | 数据结构 | 字段 |
| 上下文引用 | MQPMO | Context |
| | MQPMO | InvalidDestCount |
| | MQPMO | KnownDestCount |
| 选项 | MQPMO | 选项 |

表 611: *ImqPutMessageOptions* 交叉引用 (继续)

| 属性 | 数据结构 | 字段 |
|------------|-------|------------------|
| 记录字段 | MQPMO | PutMsgRecFields |
| 已解析队列管理器名称 | MQPMO | ResolvedQMgrName |
| 已解析队列名称 | MQPMO | ResolvedQName |
| | MQPMO | 超时 |
| | MQPMO | UnknownDestCount |
| 同步点参与 | MQPMO | 选项 |

ImqQueue 交叉引用

ImqQueue C++ 类的属性，数据结构，字段，查询和调用的交叉引用。

表 612: *ImqQueue* 交叉引用

| 属性 | 数据结构 | 字段 | 查询 | 电话 |
|------------|------|----|----------------------------|---------------|
| “回退重排队列”名称 | | | MQCA_BACKOUT_REQ_Q_NAME | MQINQ |
| 回退阈值 | | | MQIA_BACKOUT_THRESHOLD | MQINQ |
| 基本队列名称 | | | MQCA_BASE_Q_NAME | MQINQ |
| 集群名称 | | | MQCA_CLUSTER_NAME | MQINQ |
| 集群名称列表名称 | | | MQCA_CLUSTER_NAMELIST | MQINQ |
| 集群工作负载等级 | | | MQIA_CLWL_Q_RANK | MQINQ |
| 集群工作负载优先级 | | | MQIA_CLWL_Q_PRIORITY | MQINQ |
| 集群工作负载使用队列 | | | MQIA_CLWL_USEQ | MQINQ |
| 创建日期 | | | MQCA_CREATION_DATE | MQINQ |
| 创建时间 | | | MQCA_CREATION_TIME | MQINQ |
| 当前深度 | | | MQIA_CURRENT_Q_DEPTH | MQINQ |
| 缺省绑定 | | | MQIA_DEF_BIND | MQINQ |
| 缺省输入打开选项 | | | MQIA_DEF_INPUT_OPEN_OPTION | MQINQ |
| 缺省持久性 | | | MQIA_DEF_PERSISTENCE | MQINQ |
| 缺省优先级 | | | MQIA_DEF_PRIORITY | MQINQ |
| 定义类型 | | | MQIA_DEFINITION_TYPE | MQINQ |
| 深度高事件 | | | MQIA_Q_DEPTH_HIGH_EVENT | MQINQ |
| 深度上限 | | | MQIA_Q_DEPTH_HIGH_LIMIT | MQINQ |
| 深度下限事件 | | | MQIA_Q_DEPTH_LOW_EVENT | MQINQ |
| 深度下限 | | | MQIA_Q_DEPTH_LOW_LIMIT | MQINQ |
| 深度最大事件 | | | MQIA_Q_DEPTH_MAX_LIMIT | MQINQ |
| 分发表 | | | MQIA_DIST_LISTS | MQINQ 和 MQSET |

| 表 612: ImqQueue 交叉引用 (继续) | | | | |
|-----------------------------------|------|------------------|-------------------------------|---------------|
| 属性 | 数据结构 | 字段 | 查询 | 电话 |
| 动态队列名称 | MQOD | DynamicQName | | |
| 固化获取回退 | | | MQIA_HARDEN_GET_BACKOUT | MQINQ |
| 索引类型 | | | MQIA_INDEX_TYPE | MQINQ |
| 禁止获取 | | | MQIA_禁止获取 | MQINQ 和 MQSET |
| 禁止放置 | | | MQIA_禁止放入 | MQINQ 和 MQSET |
| 启动队列名称 | | | MQCA_INITIATION_Q_NAME | MQINQ |
| 最大深度 | | | MQIA_MAX_Q_DEPTH | MQINQ |
| 最大消息长度 | | | MQIA_MAX_MSG_LENGTH | MQINQ |
| 消息传递顺序 | | | MQIA_MSG_DELIVERY_SEQUENCE | MQINQ |
| 下一个分布式队列 | | | | |
| 非持久消息类 | | | MQIA_NPM_CLASS | MQINQ |
| 打开输入计数 | | | MQIA_OPEN_INPUT_COUNT | MQINQ |
| 打开输出计数 | | | MQIA_OPEN_OUTPUT_COUNT | MQINQ |
| 上一个分布式队列 | | | | |
| 流程名称 | | | MQCA_PROCESS_NAME | MQINQ |
| 队列记帐 | | | MQIA_ACCOUNTING_Q | MQINQ |
| 队列管理器名称 | MQOD | ObjectQMgrName | | |
| 队列监视 | | | MQIA_MONITORING_Q | MQINQ |
| 队列统计信息 | | | MQIA_STATISTICS_Q | MQINQ |
| 队列类型 | | | MQIA_Q_TYPE | MQINQ |
| 远程队列管理器名称 | | | MQCA_REMOTE_Q_MGR_NAME | MQINQ |
| 远程队列名称 | | | MQCA_REMOTE_Q_NAME | MQINQ |
| 已解析队列管理器名称 | MQOD | ResolvedQMgrName | | |
| 已解析队列名称 | MQOD | ResolvedQName | | |
| 保留时间间隔 | | | MQIA_RETENTION_INTERVAL | MQINQ |
| 作用域 | | | MQIA_SCOPE | MQINQ |
| 服务时间间隔 (service interval) | | | MQIA_Q_SERVICE_INTERVAL | MQINQ |
| 服务时间间隔事件 (service interval event) | | | MQIA_Q_SERVICE_INTERVAL_EVENT | MQINQ |
| 可共享性 | | | MQIA_SHAREABILITY | MQINQ |
| 存储类 (storage class) | | | MQCA_STORAGE_CLASS | MQINQ |

表 612: ImqQueue 交叉引用 (继续)

| 属性 | 数据结构 | 字段 | 查询 | 电话 |
|----------|------|----|---------------------------|---------------|
| 传输队列的名称 | | | MQCA_XMIT_Q_NAME | MQINQ |
| 触发器控制 | | | MQIA_TRIGGER_CONTROL | MQINQ 和 MQSET |
| 触发器数据 | | | MQCA_TRIGGER_DATA | MQINQ 和 MQSET |
| 触发器深度 | | | MQIA_TRIGGER_DEPTH | MQINQ 和 MQSET |
| 触发器消息优先级 | | | MQIA_TRIGGER_MSG_PRIORITY | MQINQ 和 MQSET |
| 触发器类型 | | | MQIA_TRIGGER_TYPE | MQINQ 和 MQSET |
| 用法 | | | MQIA_USAGE | MQINQ |

ImqQueueManager 交叉引用

ImqQueueManager C++ 类的属性，数据结构，字段，查询和调用的交叉引用。

| 属性 | 数据结构 | 字段 | 查询 | 电话 |
|----------------------|-------|--------------------|-------------------------------|---------|
| 记帐连接覆盖 | | | MQIA_ACCOUNTING_CONN_OVERRIDE | MQINQ |
| 记帐周期 | | | MQIA_ACCOUNTING_INTERVAL | MQINQ |
| 活动记录 | | | MQIA_ACTIVITY_录制 | MQINQ |
| 采用新的 MCA 检查 | | | MQIA_ADOPTNEWMCA_CHECK | MQINQ |
| 采用新的 MCA 类型 | | | MQIA_ADOPTNEWMCA_TYPE | MQINQ |
| 认证类型 | MQCSP | AuthenticationType | | MQCONN |
| 权限事件 | | | MQIA_AUTHORITY_EVENT | MQINQ |
| 开始选项 | MQBO | 选项 | | MQBEGIN |
| 网桥事件 | | | MQIA_BRIDGE_EVENT | MQINQ |
| 通道自动定义 | | | MQIA_CHANNEL_AUTO_DEF | MQINQ |
| 通道自动定义事件 | | | MQIA_CHANNEL_AUTO_EVENT | MQIA |
| 通道自动定义出口 | | | MQIA_CHANNEL_AUTO_EXIT | MQIA |
| 通道事件 (channel event) | | | MQIA_CHANNEL_EVENT | MQINQ |
| 通道启动程序适配器 | | | MQIA_CHINIT_ADAPTERS | MQINQ |
| 通道启动程序控制方式 | | | MQIA_CHINIT_CONTROL | MQINQ |
| 通道启动程序分派器 | | | MQIA_CHINIT_DISPATCHER | MQINQ |
| 通道启动程序跟踪自动启动 | | | MQIA_CHINIT_TRACE_AUTO_START | MQINQ |

| 属性 | 数据结构 | 字段 | 查询 | 电话 |
|-------------------------|-------|-------------------|------------------------------|--------------------------------|
| 通道启动程序跟踪表大小 | | | MQIA_CHINIT_TRACE_TABLE_SIZE | MQINQ |
| 通道监视 | | | MQIA_MONITORING_CHANNEL | MQINQ |
| 通道引用 | MQCD | ChannelType | | MQCONN |
| 通道统计 | | | MQIA_STATISTICS_CHANNEL | MQINQ |
| 字符集 | | | MQIA_CODED_CHAR_SET_ID | MQINQ |
| 集群发送方监视 | | | MQIA_MONITORING_AUTO_CLUSSDR | MQINQ |
| 集群发送方统计 | | | MQIA_STATISTICS_AUTO_CLUSSDR | MQINQ |
| 集群工作负载数据 | | | MQCA_CLUSTER_WORKLOAD_DATA | MQINQ |
| 集群工作负载出口 | | | MQCA_CLUSTER_WORKLOAD_EXIT | MQINQ |
| 集群工作负载长度 | | | MQIA_CLUSTER_WORKLOAD_LENGTH | MQINQ |
| 集群工作负载 mru | | | MQIA_CLWL_MRU_CHANNELS | MQINQ |
| 集群工作负载使用队列 | | | MQIA_CLWL_USEQ | MQINQ |
| 命令事件 (command event) | | | MQIA_COMMAND_EVENT | MQINQ |
| 命令输入队列名称 | | | MQCA_COMMAND_INPUT_Q_NAME | MQINQ |
| 命令级别 | | | MQIA_COMMAND_LEVEL | MQINQ |
| 命令服务器控制 | | | MQIA_CMD_SERVER_CONTROL | MQINQ |
| 连接选项 | MQCNO | 选项 | | MQCONN 和 MQCONN |
| 连接标识 | MQCNO | ConnectionId | | MQCONN |
| 连接状态 | | | | MQCONN , MQCONN 和 MQDISC |
| 连接标记 | MQCD | ConnTag | | MQCONN |
| 加密硬件 | MQSCO | CryptoHardware | | MQCONN |
| 死信队列名称 | | | MQCA_DEAD_LETTER_Q_NAME | MQINQ |
| 缺省传输队列名称 | | | MQCA_DEF_XMIT_Q_NAME | MQINQ |
| 分发列表 | | | MQIA_DIST_LISTS | MQINQ |
| DNS 组 | | | MQCA_DNS_GROUP | MQINQ |
| 丹 | | | MQIA_DNS_WLM | MQINQ |
| 第一个认证记录 | MQSCO | AuthInfoRecOffset | | MQCONN |
| | MQSCO | AuthInfoRecPtr | | MQCONN |
| 禁止事件 | | | MQIA_禁止事件 | MQINQ |
| IP 地址版本 | | | MQIA_IP_ADDRESS_VERSION | MQINQ |

| 属性 | 数据结构 | 字段 | 查询 | 电话 |
|--------------------------|-------|-------------------|---------------------------|--------|
| 密钥库 (key repository) | MQSCO | KeyRepository | | MQCONN |
| 密钥重置计数 | MQSCO | KeyReset 计数 | | MQCONN |
| 侦听器计时器 | | | MQIA_LISTENER_TIMER | MQINQ |
| 本地事件 | | | MQIA_LOCAL_EVENT | MQINQ |
| 记录器事件 | | | MQIA_LOGGER_EVENT | MQINQ |
| LU 组名 | | | MQCA_LU_GROUP_NAME | MQINQ |
| LU 名 | | | MQCA_LU_NAME | MQINQ |
| lu62 臂后缀 | | | MQCA_LU62_ARM_SUFFIX | MQINQ |
| lu62 通道 | | | MQIA_LU62_CHANNELS | MQINQ |
| 最大活动通道数 | | | MQIA_ACTIVE_CHANNELS | MQINQ |
| 最大通道数 | | | MQIA_MAX_CHANNELS | MQINQ |
| 最大句柄数 | | | MQIA_MAX_句柄 | MQINQ |
| 最大消息长度 | | | MQIA_MAX_MSG_LENGTH | MQINQ |
| 最高优先级 | | | MQIA_MAX_PRIORITY | MQINQ |
| 未落实消息的最大数目 | | | MQIA_MAX_UNCOMMITTED_MSGS | MQINQ |
| MQI 记帐 | | | MQIA_ACCOUNTING_MQI | MQINQ |
| MQI 统计 | | | MQIA_STATISTICS_MQI | MQINQ |
| 最大出站端口数 | | | MQIA_OUTBOUND_PORT_MAX | MQINQ |
| 出站端口最小值 | | | MQIA_OUTBOUND_PORT_MIN | MQINQ |
| 密码 | MQCSP | CSPPasswordPtr | | MQCONN |
| | MQCSP | CSPPasswordOffset | | MQCONN |
| | MQCSP | CSPPasswordLength | | MQCONN |
| 性能事件 (performance event) | | | MQIA_PERFORMANCE_EVENT | MQINQ |
| platform | | | MQIA_PLATFORM | MQINQ |
| 队列记帐 | | | MQIA_ACCOUNTING_Q | MQINQ |
| 队列监视 | | | MQIA_MONITORING_Q | MQINQ |
| 队列统计信息 | | | MQIA_STATISTICS_Q | MQINQ |
| 接收超时 | | | MQIA_RECEIVE_TIMEOUT | MQINQ |
| 接收超时最小值 | | | MQIA_RECEIVE_TIMEOUT_MIN | MQINQ |
| 接收超时类型 | | | MQIA_RECEIVE_TIMEOUT_TYPE | MQINQ |
| 远程事件 | | | MQIA_REMOTE_EVENT | MQINQ |

| 属性 | 数据结构 | 字段 | 查询 | 电话 |
|--------------|-------|---------------|--------------------------|--------|
| 存储库名称 | | | MQCA_REPOSITORY_NAME | MQINQ |
| 存储库名称列表 | | | MQCA_REPOSITORY_NAMELIST | MQINQ |
| 共享队列管理器名称 | | | MQIA_SHARED_Q_Q_MGR_NAME | MQINQ |
| ssl 事件 | | | MQIA_SSL_EVENT | MQINQ |
| 斯尔菲普斯 | | | MQIA_SSL_FIPS_REQUIRED | MQINQ |
| SSL 密钥重新设置计数 | | | MQIA_SSL_RESET_COUNT | MQINQ |
| 启动-停止事件 | | | MQIA_START_STOP_EVENT | MQINQ |
| 统计时间间隔 | | | MQIA_STATISTICS_INTERVAL | MQINQ |
| 同步点的可用性 | | | MQIA_SYNCPOINT | MQINQ |
| tcp 通道 | | | MQIA_TCP_CHANNELS | MQINQ |
| TCP 保持活动 | | | MQIA_TCP_KEEP_ALIVE | MQINQ |
| TCP 名称 | | | MQCA_TCP_NAME | MQINQ |
| TCP 堆栈类型 | | | MQIA_TCP_STACK_TYPE | MQINQ |
| 跟踪路由记录 | | | MQIA_TRACE_ROUTE_录制 | MQINQ |
| 触发器时间间隔 | | | MQIA_TRIGGER_INTERVAL | MQINQ |
| 用户标识 | MQCSP | CSPUserIdPtr | | MQCONN |
| | MQCSP | CSPUserId 偏移量 | | MQCONN |
| | MQCSP | CSPUserId 长度 | | MQCONN |

ImqReference 头交叉引用

ImqAuthenticationRecord C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|---------|-------|---------------------------|
| 目标环境 | MQRMH | DestEnv 长度, DestEnv 偏移量 |
| 目标名称 | MQRMH | DestName 长度, DestName 偏移量 |
| 实例标识 | MQRMH | ObjectInstanceId |
| 逻辑长度 | MQRMH | DataLogicalLength |
| 逻辑偏移 | MQRMH | DataLogicalOffset |
| 逻辑偏移量 2 | MQRMH | DataLogicalOffset2 |
| 引用类型 | MQRMH | ObjectType |
| 源环境 | MQRMH | SrcEnv 长度, SrcEnv 偏移量 |
| 源名称 | MQRMH | SrcName 长度, SrcName 偏移量 |

ImqTrigger 交叉引用

ImqAuthenticationRecord C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|--------|------|-------------|
| 应用程序标识 | MQTM | ApplId |
| 应用程序类型 | MQTM | ApplType |
| 环境数据 | MQTM | EnvData |
| 流程名称 | MQTM | ProcessName |
| 队列名称 | MQTM | QName |
| 触发器数据 | MQTM | TriggerData |
| 用户数据 | MQTM | UserData |

ImqWork 头交叉引用

ImqAuthenticationRecord C++ 类的属性，数据结构和字段的交叉引用。

| 属性 | 数据结构 | 字段 |
|----------------------|-------|--------------|
| 消息标记 (message token) | MQWIH | MessageToken |
| 服务名称 | MQWIH | ServiceName |
| 服务步骤 | MQWIH | ServiceStep |

ImqAuthentication 记录 C++ 类

此类封装认证信息记录 (MQAIR)，以便在执行 ImqQueueManager: :connect 方法期间用于定制 SSL 客户机连接。

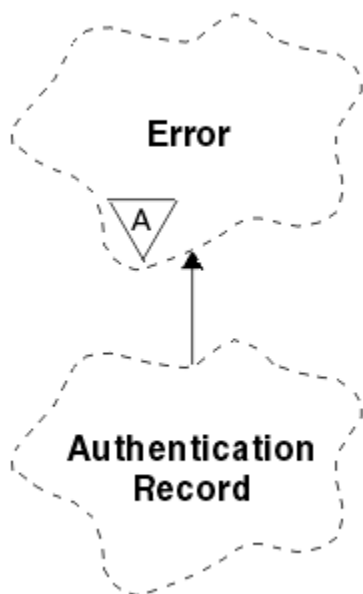


图 46: ImqAuthentication 记录类

请参阅 ImqQueueManager: :connect 方法的描述以获取更多详细信息。此类在 z/OS 平台上不可用。

- [第 1192 页的『对象属性』](#)
- [第 1192 页的『构造函数』](#)
- [第 1192 页的『对象方法 \(公用\)』](#)
- [第 1193 页的『对象方法 \(protected\)』](#)

对象属性

连接名称

与 LDAP CRL 服务器的连接的名称。这是 IP 地址或 DNS 名称，后跟 (可选) 括号中的端口号。

连接引用

对 `ImqQueueManager` 对象的引用，该对象提供与 (本地) 队列管理器的必需连接。初始值为零。请勿将此与用于标识指定队列的队列管理器 (可能是远程队列) 的队列管理器名称混淆。

下一个认证记录

此类的下一个对象没有特殊顺序，具有与此对象相同的 **连接引用**。初始值为零。

密码

为 LDAP CRL 服务器的连接认证提供的密码。

先前认证记录

此类的先前对象 (无特定顺序) 具有与此对象相同的 **连接引用**。初始值为零。

type

记录中包含的认证信息的类型。

用户名

为 LDAP CRL 服务器的授权提供的用户标识。

构造函数

`ImqAuthenticationRecord ()`;

缺省构造函数。

对象方法 (公用)

`void operator = (const ImqAuthenticationRecord & 空气);`

从 *air* 复制实例数据，替换现有实例数据。

`const ImqString & connectionName () const ;`

返回 **连接名称**。

`void setConnectionName (const ImqString & 名);`

设置 **连接名称**。

`void setConnectionName (const char * name = 0);`

设置 **连接名称**。

`ImqQueueManager * connectionReference ()` 康斯特;

返回 **连接引用**。

`void setConnectionReference (ImqQueueManager & 经理);`

设置 **连接引用**。

`void setConnectionReference (ImqQueueManager * manager = 0);`

设置 **连接引用**。

`void copyOut (MQAIR * pAir);`

将实例数据复制到 *pAir*，以替换现有实例数据。这可能涉及分配从属存储器。

`空清除 (MQAIR * pAir);`

清除结构并释放 *pAir* 引用的从属存储器。

`ImqAuthenticationRecord * nextAuthenticationRecord ()` 康斯特;

返回 **下一个认证记录**。

`const ImqString & password () const ;`

返回 **password**。

`void setPassword (const ImqString & 密码);`

设置 **password**。

`void setPassword (const char * password = 0);`

设置 **password**。

ImqAuthentication 记录 * **previousAuthenticationRecord** () 康斯特;

返回 先前认证记录。

MQLONG 类型 () 康斯特;

返回 **type**。

void setType (const MQLONG type);

设置 **type**。

const ImqString & userName () const ;

返回 用户名。

void setUsername (const ImqString & 名);

设置 用户名。

void setUsername (const char * name = 0);

设置 用户名。

对象方法 (protected)

void setNextAuthenticationRecord (ImqAuthenticationRecord * pAir = 0);

设置 下一个认证记录。

注意: 仅当您确定此功能不会破坏认证记录列表时, 才使用此功能。

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * pAir = 0);

设置 先前认证记录。

注意: 仅当您确定此功能不会破坏认证记录列表时, 才使用此功能。

ImqBinary C++ 类

此类封装可用于 ImqMessage 记帐标记, 相关标识和 消息标识 值的二进制字节数组。它允许简单的分配, 复制和比较。

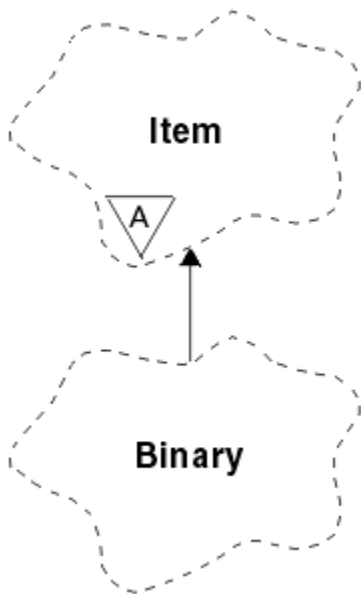


图 47: ImqBinary 类

- [第 1194 页的『对象属性』](#)
- [第 1194 页的『构造函数』](#)
- [第 1194 页的『重载的 ImqItem 方法』](#)
- [第 1194 页的『对象方法 \(公用\)』](#)
- [第 1195 页的『对象方法 \(protected\)』](#)

对象属性

data

二进制数据的字节数组。初始值为空。

数据长度

字节数。初始值为零。

数据指针

data 的第一个字节的地址。初始值为零。

构造函数

ImqBinary();

缺省构造函数。

ImqBinary(const ImqBinary & 二进制);

复制构造函数。

ImqBinary(const void * data, const size_t length);

从数据复制 *length* 个字节。

重载的 ImqItem 方法

虚拟 **ImqBoolean copyOut(ImqMessage & 消息);**

将 **数据** 复制到消息缓冲区，以替换任何现有内容。将 **msg 格式** 设置为 MQFMT_NONE。

请参阅 ImqItem 类方法描述以获取更多详细信息。

虚拟 **ImqBoolean pasteIn(ImqMessage & 消息);**

通过从消息缓冲区传输剩余数据 (替换现有 **数据**) 来设置 **数据**。

要成功，ImqMessage **格式** 必须为 MQFMT_NONE。

请参阅 ImqItem 类方法描述以获取更多详细信息。

对象方法 (公用)

void 运算符 = (const ImqBinary & 二进制);

从 *binary* 复制字节。

ImqBoolean 运算符 == (const ImqBinary & 二进制);

将此对象与 *binary* 进行比较。如果不等于，那么返回 FALSE，否则返回 TRUE。如果对象具有相同的 **数据长度** 且字节匹配，那么这些对象相等。

ImqBoolean copyOut(void * buffer, const size_t length, const char pad = 0);

将最多 *length* 个字节从 **数据指针** 复制到缓冲区。如果 **数据长度** 不足，那么将使用 *pad* 字节填充 *buffer* 中的剩余空间。如果 *length* 也为零，那么 *buffer* 可以为零。*length* 不得为负数。如果成功，将返回 TRUE。

size_t dataLength() const ;

返回 **数据长度**。

ImqBoolean setDataLength(const size_t length);

设置 **数据长度**。如果由于此方法而更改了 **data length**，那么对象中的数据将处于未初始化状态。如果成功，将返回 TRUE。

void * dataPointer() const ;

返回 **数据指针**。

ImqBoolean isNull() const ;

如果 **data length** 为零，或者如果所有 **data** 字节都为零，那么返回 TRUE。否则，将返回 FALSE。

ImqBoolean set(const void * buffer, const size_t length);

从 *buffer* 复制 *length* 字节。如果成功，将返回 TRUE。

对象方法 (protected)

`void clear();`

将 **数据长度** 减小为零。

原因码

- MQRC_NO_BUFFER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_INCONSISTENT_FORMAT

ImqCache C++ 类

使用此类来保存或编组内存中的数据。

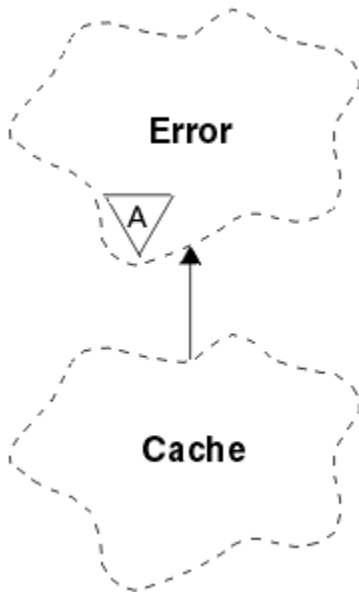


图 48: *ImqCache* 类

使用此类来保存或编组内存中的数据。您可以指定固定大小的内存缓冲区，或者系统可以自动提供灵活的内存量。此类与 [第 1179 页的『ImqCache 交叉引用』](#) 中列出的 MQI 调用相关。

- [第 1195 页的『对象属性』](#)
- [第 1196 页的『构造函数』](#)
- [第 1196 页的『对象方法 \(公用\)』](#)
- [第 1197 页的『原因码』](#)

对象属性

自动缓冲区

指示缓冲区内存是由系统自动管理 (TRUE) 还是由用户提供 (FALSE)。它最初设置为 TRUE。

未直接设置此属性。它是使用 `useEmptyBuffer` 或 `useFullBuffer` 方法间接设置的。

如果提供了用户存储器，那么此属性为 FALSE，缓冲区内存无法增长，并且可能会发生缓冲区溢出错误。缓冲区的地址和长度保持不变。

如果未提供用户存储器，那么此属性为 TRUE，并且缓冲区内存可以递增增长以容纳任意数量的消息数据。但是，当缓冲区增大时，缓冲区的地址可能会更改，因此使用 **缓冲区指针** 和 **数据指针** 时请小心。

缓冲区长度

缓冲区中的内存字节数。初始值为零。

缓冲区指针

缓冲区内存的地址。初始值为空。

数据长度

数据指针之后的字节数。这必须等于或小于消息长度。初始值为零。

数据偏移量

数据指针之前的字节数。这必须等于或小于消息长度。初始值为零。

数据指针

要写入或从下一个缓冲区中读取的部分的地址。初始值为空。

消息长度

缓冲区中重要数据的字节数。初始值为零。

构造函数

ImqCache();

缺省构造函数。

ImqCache(const ImqCache & 高速缓存);

复制构造函数。

对象方法 (公用)

void 运算符 = (const ImqCache & 高速缓存);

将最多 **message length** 字节的数据从 *cache* 对象复制到该对象。如果 **自动缓冲区** 为 **FALSE**，那么 **缓冲区长度** 必须已足以容纳复制的数据。

ImqBoolean automaticBuffer() const ;

返回 **自动缓冲区** 值。

size_t bufferSize() const ;

返回 **缓冲区长度**。

char * bufferPointer() const ;

返回 **缓冲区指针**。

void clearMessage();

将 **消息长度** 和 **数据偏移量** 设置为零。

size_t dataLength() const ;

返回 **数据长度**。

size_t dataOffset() const ;

返回 **数据偏移量**。

ImqBoolean setDataOffset(const size_t offset);

设置 **数据偏移量**。如果需要，将增大 **消息长度**，以确保其不小于 **数据偏移量**。如果成功，此方法将返回 **TRUE**。

char * dataPointer() const ;

返回 **数据指针** 的副本。

size_t messageLength() const ;

返回 **消息长度**。

ImqBoolean setMessageLength(const size_t length);

设置 **消息长度**。如果需要，请增大 **缓冲区长度**，以确保 **消息长度** 不大于 **缓冲区长度**。必要时减小 **数据偏移量**，以确保其不大于 **消息长度**。如果成功，将返回 **TRUE**。

ImqBoolean moreBytes(const size_t bytes-required);

确保在 **数据指针** 与缓冲区末尾之间提供 *bytes-required* 更多字节 (用于写入)。如果成功，将返回 **TRUE**。

如果 **自动缓冲区** 为 **TRUE**，那么将根据需要获取更多内存; 否则，**缓冲区长度** 必须已足够。

ImqBoolean 读(连接大小 (t) 长度, char * & 外部缓冲区);

将 *length* 字节从缓冲区从 **数据指针** 位置开始复制到 *external-buffer*。复制数据后, **数据偏移量** 将增大 *length*。如果成功, 此方法将返回 TRUE。

ImqBoolean resizeBuffer(const size_t length);

更改 **缓冲区长度**, 前提是 **自动缓冲区** 为 TRUE。这是通过重新分配缓冲区内存来实现的。将现有缓冲区中最多 **消息长度** 字节的数据复制到新缓冲区。复制的最大数目为 *length* 字节。**缓冲区指针** 已更改。**消息长度** 和 **数据偏移量** 将尽可能在新缓冲区的范围内保留。如果成功, 那么返回 TRUE, 如果 **自动缓冲区** 为 FALSE, 那么返回 FALSE。

注: 如果系统资源存在任何问题, 那么此方法可能会失败, 并返回 MQRC_STORAGE_NOT_AVAILABLE。

ImqBoolean useEmptyBuffer(const char * external-buffer, const size_t length);

标识空的用户缓冲区, 将 **缓冲区指针** 设置为指向 *external-buffer*, 将 **缓冲区长度** 设置为 *length*, 将 **message length** 设置为零。执行 **clearMessage**。如果缓冲区已完全准备好数据, 请改为使用 **useFullBuffer** 方法。如果缓冲区部分准备了数据, 请使用 **setMessageLength** 方法来指示正确的数量。如果成功, 此方法将返回 TRUE。

此方法可用于标识固定内存量, 如前所述 (*external-buffer* 不为空且 *length* 为非零), 在此情况下, **automatic buffer** 设置为 FALSE, 或者可用于还原为系统管理的灵活内存 (*external-buffer* 为空且 *length* 为零), 在此情况下, **automatic buffer** 设置为 TRUE。

ImqBoolean useFullBuffer(const char * externalBuffer, const size_t length);

对于 **useEmptyBuffer**, 除了 **message length** 设置为 *length*。如果成功, 将返回 TRUE。

ImqBoolean write(const size_t length, const char * external-buffer);

将 *length* 个字节从 *external-buffer* 复制到缓冲区中, 从 **数据指针** 位置开始。复制数据后, **data offset** 将增大 *length*, 如果需要, 将增大 **message length**, 以确保它不小于新的 **data offset** 值。如果成功, 此方法将返回 TRUE。

如果 **自动缓冲区** 为 TRUE, 那么保证足够的内存量; 否则, 最终 **数据偏移量** 不得超过 **缓冲区长度**。

原因码

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_截断
- MQRC_IN 区内的缓冲区
- MQRC_IN 有数据
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_ZERO_LENGTH

ImqChannel C++ 类

此类封装了通道定义 (MQCD), 以便在执行 Manager: :connect 方法期间用于定制客户机连接。

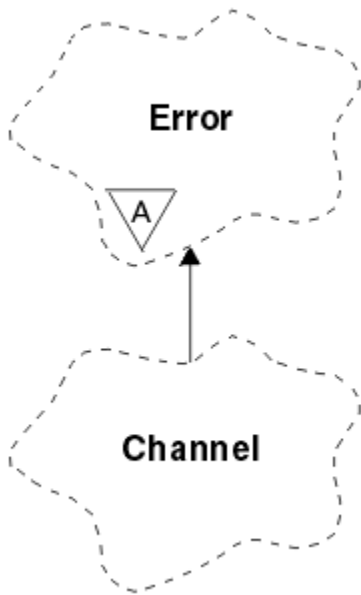


图 49: *ImqChannel* 类

有关更多详细信息，请参阅 `Manager: :connect` 方法和 样本程序 `HELLO WORLD (imqworld.cpp)` 的描述。并非所有列出的方法都适用于所有平台；请参阅 [MQSC 引用](#) 中的 `DEFINE CHANNEL` 和 `ALTER CHANNEL` 命令的描述以获取更多详细信息。*ImqChannel* 类在 z/OS 上不受支持。

- [第 1198 页的『对象属性』](#)
- [第 1199 页的『构造函数』](#)
- [第 1199 页的『对象方法 \(公用\)』](#)
- [第 1202 页的『原因码』](#)

对象属性

批处理脉动信号

检查远程通道是否处于活动状态之间的毫秒数。初始值为 0。

通道名称

通道的名称。初始值为空。

连接名称

连接的名称。例如，主机的 IP 地址。初始值为空。

头压缩

通道所支持的头数据压缩技术列表。初始值全部设置为 `MQCOMPRESS_NOT_AVAILABLE`。

脉动信号间隔

两次检查连接是否仍在工作的秒数。初始值为 300。

保持活动时间间隔

传递到通信堆栈的秒数，用于指定通道的保持活动计时。初始值为 `MQKAI_AUTO`。

本地地址

通道的本地通信地址。

最大消息长度

通道在单个通信中支持的最大消息长度。初始值为 4 194 304。

消息压缩

通道支持的消息数据压缩技术的列表。初始值全部设置为 `MQCOMPRESS_NOT_AVAILABLE`。

方式名

方式的名称。初始值为空。

password

为连接认证提供的密码。初始值为空。

接收出口计数

接收出口的数量。初始值为零。此属性是只读的。

接收出口名称

接收出口的名称。

接收用户数据

与接收出口关联的数据。

安全出口名

要在连接的服务器端调用的安全出口的名称。初始值为空。

安全用户数据

要传递到安全出口的数据。初始值为空。

发送出口计数

发送出口数。初始值为零。此属性是只读的。

发送出口名称

发送出口的名称。

发送用户数据

与发送出口关联的数据。

SSL CipherSpec

CipherSpec, 用于 SSL。

SSL 客户机认证类型

用于 SSL 的客户机认证类型。

SSL 对等名称

用于 SSL 的对等名称。

事务程序名

事务程序的名称。初始值为空。

传输类型

连接的传输类型。初始值为 MQXPT_LU62。

用户标识

为授权提供的用户标识。初始值为空。

构造函数**ImqChannel();**

缺省构造函数。

ImqChannel(const ImqChannel & 通道);

复制构造函数。

对象方法 (公用)**void operator = (const ImqChannel & 通道);**

从 通道复制实例数据, 替换任何现有实例数据。

MQLONG batchHeartBeat () 康斯特;

返回 batch heart-beat。

ImqBoolean setBatchHeartBeat(const MQLONG 脉动信号 = 0L);

设置 批处理脉动信号。如果成功, 此方法将返回 TRUE。

ImqString channelName() 康斯特;

返回 通道名称。

ImqBoolean setChannelName(const char * name = 0);

设置 通道名称。如果成功, 此方法将返回 TRUE。

ImqString connectionName() 康斯特;

返回 连接名称。

ImqBoolean setConnectionName (const char * name = 0);

设置 连接名称。 如果成功，此方法将返回 TRUE。

size_t headerCompressionCount () 康斯特;

返回受支持的头数据压缩技术计数。

ImqBoolean headerCompression(const size_t count , MQLONG compress []) 康斯特;

返回 **compress** 中受支持的头数据压缩技术的副本。 如果成功，此方法将返回 TRUE。

ImqBoolean setHeader 压缩 (const size_t count , const MQLONG compress []);

将受支持的头数据压缩技术设置为 压缩。

将受支持的头数据压缩技术计数设置为 **count**。

如果成功，此方法将返回 TRUE。

MQLONG heartBeat 时间间隔 () 康斯特;

返回 脉动信号间隔。

ImqBoolean setHeartBeatInterval(const MQLONG interval = 300L);

设置 脉动信号间隔。 如果成功，此方法将返回 TRUE。

MQLONG keepAlive 时间间隔 () 康斯特;

返回 保持活动时间间隔。

ImqBoolean setKeepAliveInterval(const MQLONG interval = MQKAI_AUTO);

设置 保持活动时间间隔。 如果成功，此方法将返回 TRUE。

ImqString localAddress() const;

返回 本地地址。

ImqBoolean setLocalAddress (const char * address = 0);

设置 本地地址。 如果成功，此方法将返回 TRUE。

MQLONG maximumMessage 长度 () 康斯特;

返回 最大消息长度。

ImqBoolean setMaximumMessageLength(const MQLONG length = 4194304L);

设置 最大消息长度。 如果成功，此方法将返回 TRUE。

size_t messageCompressionCount () 康斯特;

返回受支持的消息数据压缩技术计数。

ImqBoolean messageCompression(const size_t count , MQLONG compress []) const;

返回 **compress** 中受支持的消息数据压缩技术的副本。 如果成功，此方法将返回 TRUE。

ImqBoolean setMessage 压缩 (const size_t count , const MQLONG compress []);

设置要压缩的受支持消息数据压缩技术。

将受支持的消息数据压缩技术计数设置为计数。

如果成功，此方法将返回 TRUE。

ImqString modeName() 康斯特;

返回 mode name。

ImqBoolean setModeName (const char * name = 0);

设置 方式名。 如果成功，此方法将返回 TRUE。

ImqString password () 康斯特;

返回 password。

ImqBoolean setPassword(const char * password = 0);

设置 password。 如果成功，此方法将返回 TRUE。

size_t receiveExit 计数 () 康斯特;

返回 接收出口计数。

ImqString receiveExit 名称 ();

返回第一个 **接收出口名称**(如果有)。如果 **接收出口计数** 为零, 那么将返回空字符串。

ImqBoolean receiveExit 名称 (const size_t count, ImqString * names []);

返回 *names* 中 **receive exit names** 的副本。将超过 **receive exit count** 的任何 *names* 设置为空字符串。如果成功, 此方法将返回 TRUE。

ImqBoolean setReceiveExitName(const char * name = 0);

将 **接收出口名称** 设置为单个名称。 *name* 可以为空或空。将 **接收出口计数** 设置为 1 或零。清除 **接收用户数据**。如果成功, 此方法将返回 TRUE。

ImqBoolean setReceiveExitNames(const size_t count, const char * names []);

将 **receive exit names** 设置为 *names*。单个 *names* 值不得为空或为空。将 **接收出口计数** 设置为 *count*。清除 **接收用户数据**。如果成功, 此方法将返回 TRUE。

ImqBoolean setReceiveExitNames(const size_t count, const ImqString * names []);

将 **receive exit names** 设置为 *names*。单个 *names* 值不得为空或为空。将 **接收出口计数** 设置为 *count*。清除 **接收用户数据**。如果成功, 此方法将返回 TRUE。

ImqString receiveUser 数据 ();

返回第一个 **接收用户数据** 项(如果有)。如果 **接收出口计数** 为零, 那么返回空字符串。

ImqBoolean receiveUser 数据 (const size_t count, ImqString * data []);

返回 *数据* 中 **接收用户数据** 项的副本。将超过 **接收出口计数** 的任何 *数据* 设置为空字符串。如果成功, 此方法将返回 TRUE。

ImqBoolean setReceiveUserData(const char * data = 0);

将 **接收用户数据** 设置为单个项 *数据*。如果 *data* 不为空, 那么 **接收出口计数** 必须至少为 1。如果成功, 此方法将返回 TRUE。

ImqBoolean setReceiveUserData(const size_t count, const char * data []);

将 **接收用户数据** 设置为 *数据*。 *count* 不得大于 **receive exit count**。如果成功, 此方法将返回 TRUE。

ImqBoolean setReceiveUserData(const size_t count, const ImqString * data []);

将 **接收用户数据** 设置为 *数据*。 *count* 不得大于 **receive exit count**。如果成功, 此方法将返回 TRUE。

ImqString securityExit 名称 () 康斯特;

返回 **安全出口名称**。

ImqBoolean setSecurityExitName(const char * name = 0);

设置 **安全出口名称**。如果成功, 此方法将返回 TRUE。

ImqString securityUser 数据 () 康斯特;

返回 **安全用户数据**。

ImqBoolean setSecurityUserData(const char * data = 0);

设置 **安全用户数据**。如果成功, 此方法将返回 TRUE。

size_t sendExit 计数 () 康斯特;

返回 **发送出口计数**。

ImqString sendExitName ();

返回第一个 **发送出口名称**(如果有)。如果 **发送出口计数** 为零, 那么返回空字符串。

ImqBoolean sendExit 名称 (const size_t count, ImqString * names []);

返回 *names* 中 **send exit names** 的副本。将超过 **send exit count** 的任何 *names* 设置为空字符串。如果成功, 此方法将返回 TRUE。

ImqBoolean setSendExitName(const char * name = 0);

将 **发送出口名称** 设置为单个名称。 *name* 可以为空或空。将 **发送出口计数** 设置为 1 或零。清除 **发送用户数据**。如果成功, 此方法将返回 TRUE。

ImqBoolean setSendExitNames(const size_t count, const char * names []);

将 **send exit names** 设置为 *names*。单个 *names* 值不得为空或为空。将 **发送出口计数** 设置为 *count*。清除 **发送用户数据**。如果成功, 此方法将返回 TRUE。

ImqBoolean setSendExitNames(const size_t count, const ImqString * names []);

将 **send exit names** 设置为 *names*。单个 *names* 值不得为空或为空。将 **发送出口计数** 设置为 *count*。清除 **发送用户数据**。如果成功, 此方法将返回 TRUE。

ImqString sendUserData ();

返回第一个 **发送用户数据** 项 (如果有)。如果 **发送出口计数** 为零, 那么返回空字符串。

ImqBoolean sendUserData (const size_t count, ImqString * data []);

返回 *data* 中 **send user data** 项的副本。将超过 **发送出口计数** 的任何 **数据** 设置为空字符串。如果成功, 此方法将返回 TRUE。

ImqBoolean setSendUserData(const char * data = 0);

将 **发送用户数据** 设置为单个项 **数据**。如果 *data* 不为空, 那么 **send exit count** 必须至少为 1。如果成功, 此方法将返回 TRUE。

ImqBoolean setSendUserData(const size_t count, const char * data []);

将 **发送用户数据** 设置为 **数据**。*count* 必须不大于 **send exit count**。如果成功, 此方法将返回 TRUE。

ImqBoolean setSendUserData(const size_t count, const ImqString * data []);

将 **发送用户数据** 设置为 **数据**。*count* 必须不大于 **send exit count**。如果成功, 此方法将返回 TRUE。

ImqString sslCipher 规范 () 康斯特;

返回 SSL 密码规范。

ImqBoolean setSslCipherSpecification(const char * name = 0);

设置 SSL 密码规范。如果成功, 此方法将返回 TRUE。

MQLONG sslClient 认证 () 康斯特;

返回 SSL 客户机认证类型。

ImqBoolean setSslClientAuthentication(const MQLONG auth = MQSCA_REQUIRED);

设置 SSL 客户机认证类型。如果成功, 此方法将返回 TRUE。

ImqString sslPeer 名称 () 康斯特;

返回 SSL 对等名称。

ImqBoolean setSslPeerName(const char * name = 0);

设置 SSL 对等名称。如果成功, 此方法将返回 TRUE。

ImqString transactionProgram 名称 () 康斯特;

返回 **事务程序名**。

ImqBoolean setTransactionProgramName(const char * name = 0);

设置 **事务程序名**。如果成功, 此方法将返回 TRUE。

MQLONG transportType() 康斯特;

返回 **传输类型**。

ImqBoolean setTransportType (const MQLONG type = MQXPT_LU62);

设置 **传输类型**。如果成功, 此方法将返回 TRUE。

ImqString userId() 康斯特;

返回 **用户标识**。

ImqBoolean setUserId (const char * id = 0);

设置 **用户标识**。如果成功, 此方法将返回 TRUE。

原因码

- MQRC_DATA_LENGTH_ERROR
- MQRC_ITEM_COUNT_ERROR
- MQRC_NULL_POINTER
- MQRC_SOURCE_BUFFER_ERROR

ImqCICSBridge 头 C++ 类

此类封装 MQCIH 数据结构的特定功能部件。

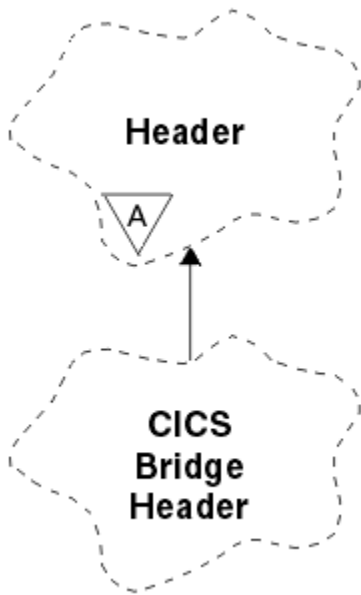


图 50: *ImqCICSBridge* 头类

此类的对象由通过 WebSphere MQ for z/OS 将消息发送到 CICS 网桥的应用程序使用。

- [第 1203 页的『对象属性』](#)
- [第 1205 页的『构造函数』](#)
- [第 1205 页的『重载的 *ImqItem* 方法』](#)
- [第 1206 页的『对象方法 \(公用\)』](#)
- [第 1208 页的『对象数据 \(受保护\)』](#)
- [第 1208 页的『原因码』](#)
- [第 1208 页的『返回码』](#)

对象属性

ADS 描述符

发送/接收 ADS 描述符。这是使用 MQCADSD_NONE 设置的。初始值为 MQCADSD_NONE。可以使用以下附加值：

- MQCADSD_NONE
- MQCADSD_SEND
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

辅助操作请求标识

AID 键。该字段的长度必须为 MQ_ATTENTION_ID_LENGTH。

鉴别符 (authenticator)

RACF 密码或通行票。初始值包含空白，长度为 MQ_AUTHENTICATOR_LENGTH。

网桥异常终止代码

网桥异常终止代码，长度为 MQ_ABEND_CODE_LENGTH。初始值为四个空白字符。此字段中返回的值取决于返回码。请参阅[第 1208 页的表 614](#) 以获取更多详细信息。

网桥取消代码

网桥异常终止事务代码。字段是保留的，必须包含空白，且长度为 MQ_CANCEL_CODE_LENGTH。

网桥完成代码

完成代码，可以包含 WebSphere MQ 完成代码或 CICS EIBRESP 值。该字段的初始值为 MQCC_OK。此字段中返回的值取决于返回码。请参阅[第 1208 页的表 614](#) 以获取更多详细信息。

网桥错误偏移量

网桥错误偏移量。初始值为零。此属性是只读的。

网桥原因码

原因码。此字段可以包含 WebSphere MQ 原因或 CICS EIBRESP2 值。该字段的初始值为 MQRC_NONE。此字段中返回的值取决于返回码。请参阅第 1208 页的表 614 以获取更多详细信息。

网桥返回码

来自 CICS 网桥的返回码。初始值为 MQCRC_OK。

会话式任务

任务是否可以会话式任务。初始值为 MQCCT_NO。可以使用以下附加值：

- MQCCT_YES
- MQCCT_NO

光标位置

光标位置。初始值为零。

设施保留时间

CICS 网桥设施发布时间。

工具, 例如

终端仿真属性。该字段的长度必须为 MQ_FACILITY_LIKE_LENGTH。

设施令牌

BVT 令牌值。该字段的长度必须为 MQ_FACILITY_LENGTH。初始值为 MQCFAC_NONE。

函数

函数，可以包含 WebSphere MQ 调用名称或 CICS EIBFN 函数。该字段的初始值为 MQCFUNC_NONE，长度为 MQ_FUNCTION_LENGTH。此字段中返回的值取决于返回码。请参阅第 1208 页的表 614 以获取更多详细信息。

当 **function** 包含 WebSphere MQ 调用名称时，可以使用以下附加值：

- MQCFUNC_MQCONN
- MQCFUNC_MQGET
- MQCFUNC_MQINQ
- MQCFUNC_NONE
- MQCFUNC_MQOPEN
- MQCFUNC_PUT
- MQCFUNC_MQPUT1

获取等待时间间隔

等待 CICS 网桥任务发出的 MQGET 调用的时间间隔。初始值为 MQCGWI_DEFAULT。仅当 **uow control** 的值为 MQCUOWC_FIRST 时，该字段才适用。可以使用以下附加值：

- MQCGWI_DEFAULT
- MQWI_UNLIMITED

链接类型

链接类型。初始值为 MQCLT_PROGRAM。可以使用以下附加值：

- MQCLT_PROGRAM
- MQCLT_TRANSACTION

下一个事务标识

要连接的下一个事务的标识。该字段的长度必须为 MQ_TRANSACTION_ID_LENGTH。

输出数据长度

COMMAREA 数据长度。初始值为 MQCODL_AS_INPUT。

回复格式

应答消息的格式名。初始值为 MQFMT_NONE，长度为 MQ_FORMAT_LENGTH。

启动代码

事务开始代码。该字段的长度必须为 MQ_START_CODE_LENGTH。初始值为 MQCSC_NONE。可以使用以下附加值:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_NONE

任务结束状态

任务结束状态。初始值为 MQCTES_NOSYNC。可以使用以下附加值:

- MQCTES_COMMIT
- MQCTES_BACKOUT
- MQCTES_ENDTASK
- MQCTES_NOSYNC

事务标识

要连接的事务的标识。初始值必须包含空白, 并且长度必须为 MQ_TRANSACTION_ID_LENGTH。仅当 **uow control** 具有值 MQCUOWC_FIRST 或 MQCUOWC_ONLY 时, 该字段才适用。

UOW 控件

UOW 控制。初始值为 MQCUOWC_ONLY。可以使用以下附加值:

- MQCUOWC_FIRST
- MQCUOWC_中间件
- MQCUOWC_LAST
- MQCUOWC_ONLY
- MQCUOWC_COMMIT
- MQCUOWC_BACKOUT
- MQCUOWC_CONTINUE

version

MQCIH 版本号。初始值为 MQCIH_VERSION_2。唯一受支持的其他值为 MQCIH_VERSION_1。

构造函数

ImqCICSBridgeHeader ();

缺省构造函数。

ImqCICSBridge 头 (const ImqCICSBridge 头和 头);

复制构造函数。

重载的 ImqItem 方法

virtual ImqBoolean copyOut(ImqMessage & 消息);

在开始时将 MQCIH 数据结构插入到消息缓冲区中, 进一步移动现有消息数据, 并将消息格式设置为 MQFMT_CICS。

请参阅父类方法描述以获取更多详细信息。

virtual ImqBoolean pasteIn(ImqMessage & 消息);

从消息缓冲区读取 MQCIH 数据结构。要成功, *msg* 对象的编码必须是 MQENC_NATIVE。使用 MQGMO_CONVERT 检索到 MQENC_NATIVE 的消息。要成功, *ImqMessage* 格式必须为 MQFMT_CICS。

请参阅父类方法描述以获取更多详细信息。

对象方法 (公用)

void operator = (const ImqCICSBridgeHeader & 标题);

从 头复制实例数据，替换现有实例数据。

MQLONG ADSDescriptor () 康斯特;

返回 ADS 描述符的副本。

void setADSDescriptor(const MQLONG 描述符 = MQCADSD_NONE);

设置 ADS 描述符。

ImqString attentionIdentifier() 康斯特;

返回 辅助操作请求标识的副本，使用尾部空格填充此副本的长度为 MQ_ATTENTION_ID_LENGTH。

void setAttentionIdentifier (const char * data = 0);

将使用尾部空格填充的 辅助操作请求标识设置为长度 MQ_ATTENTION_ID_LENGTH。如果未提供 data，请将 辅助操作请求标识 重置为初始值。

ImqString 鉴别符 () 康斯特;

返回 authenticator 的副本，以尾部空格填充至长度 MQ_AUTHENTICATOR_LENGTH。

void setAuthenticator(const char * data = 0);

将使用尾部空格填充的 authenticator 设置为长度 MQ_AUTHENTICATOR_LENGTH。如果未提供数据，请将 authenticator 重置为初始值。

ImqString bridgeAbend 代码 () 康斯特;

返回 bridge abend code 的副本，用尾部空格填充此副本的长度为 MQ_ABEND_CODE_LENGTH。

ImqString bridgeCancel 代码 () 康斯特;

返回 网桥取消代码的副本，并以尾部空格填充以长度 MQ_CANCEL_CODE_LENGTH。

void setBridgeCancelCode(const char * data = 0);

将使用尾部空格填充的 网桥取消代码设置为长度 MQ_CANCEL_CODE_LENGTH。如果未提供数据，请将 网桥取消代码 重置为初始值。

MQLONG bridgeCompletion 代码 () 康斯特;

返回 网桥完成代码的副本。

MQLONG bridgeErrorOffset () 康斯特;

返回 bridge error offset 的副本。

MQLONG bridgeReason 代码 () 康斯特;

返回 网桥原因码的副本。

MQLONG bridgeReturn 代码 () 康斯特;

返回 网桥返回码。

MQLONG conversationalTask() 康斯特;

返回 会话式任务的副本。

void setConversationalTask (const MQLONG task = MQCCT_NO);

设置 会话式任务。

MQLONG cursorPosition() 康斯特;

返回 光标位置的副本。

void setCursorPosition (const MQLONG position = 0);

设置 光标位置。

MQLONG facilityKeep 时间 () 康斯特;

返回 工具保留时间的副本。

void setFacilityKeepTime(const MQLONG time = 0);

设置 设施保留时间。

ImqString facilityLike() 康斯特;

返回 工具 (例如) 的副本，用尾部空格填充到长度 MQ_FACILITY_LIKE_LENGTH。

void setFacilityLike (const char * name = 0);

设置 工具 (例如)，用尾部空格填充到长度 MQ_FACILITY_LIKE_LENGTH。如果未提供 name，请重置工具，例如 初始值。

ImqBinary facilityToken() 康斯特;

返回 工具令牌的副本。

ImqBoolean setFacilityToken(const ImqBinary & 令牌);

设置 设施令牌。令牌的数据长度必须为零或 MQ_FACILITY_LENGTH。如果成功，将返回 TRUE。

void setFacilityToken(const MQBYTE8 token = 0);

设置 设施令牌。token 可以为零，这与指定 MQCFAC_NONE 相同。如果 token 非零，那么它必须处理二进制数据的 MQ_FACILITY_LENGTH 字节。使用预定义值 (例如 MQCFAC_NONE) 时，您可能需要进行强制类型转换以确保签名匹配。例如，(MQBYTE *) MQCFAC_NONE。

ImqString 函数 () 康斯特;

返回 function 的副本，以尾部空格填充到长度 MQ_FUNCTION_LENGTH。

MQLONG getWait 时间间隔 () 康斯特;

返回 get wait interval 的副本。

void setGetWaitInterval(const MQLONG interval = MQCGWI_DEFA

设置 获取等待时间间隔。

MQLONG linkType() 康斯特;

返回 链接类型的副本。

void setLinkType(const MQLONG type = MQCLT_PROGRAM);

设置 链接类型。

ImqString nextTransactionIdentifier () 康斯特;

返回 下一个事务标识 数据的副本，以尾部空格填充，长度为 MQ_TRANSACTION_ID_LENGTH。

MQLONG outputData 长度 () 康斯特;

返回 输出数据长度的副本。

void setOutputDataLength(const MQLONG length = MQCODL_AS_INPUT);

设置 输出数据长度。

ImqString replyTo 格式 () 康斯特;

返回 reply-to format 名称的副本，以尾部空格填充到长度 MQ_FORMAT_LENGTH。

void setReplyToFormat(const char * name = 0);

设置 应答格式，用尾部空格填充到长度 MQ_FORMAT_LENGTH。如果未提供 name，请将 reply-to format 重置为初始值。

ImqString startCode() 康斯特;

返回 启动代码的副本，填充了长度为 MQ_START_CODE_LENGTH 的尾部空格。

void setStartCode(const char * data = 0);

将 启动代码 数据 (用尾部空格填充) 设置为长度 MQ_START_CODE_LENGTH。如果未提供 数据，请将 启动代码 重置为初始值。

MQLONG taskEnd 状态 () 康斯特;

返回 任务结束状态的副本。

ImqString transactionIdentifier() 康斯特;

返回 事务标识 数据的副本，以尾部空格填充到长度 MQ_TRANSACTION_ID_LENGTH。

void setTransactionIdentifier(const char * data = 0);

设置 事务标识，用尾部空格填充到长度 MQ_TRANSACTION_ID_LENGTH。如果未提供 数据，请将 事务标识 重置为初始值。

MQLONG UOWControl () 康斯特;

返回 UOW 控件的副本。

void setUOWControl(const MQLONG control = MQCUOWC_ONLY);

设置 UOW 控制。

MQLONG 版本 () 康斯特;

返回 version 数字。

ImqBoolean setVersion(const MQLONG version = MQCIH_VERSION_2);

设置 version 编号。如果成功，将返回 TRUE。

对象数据 (受保护)

MQLONG olVersion

分配给 *opcih* 的存储器中可容纳的最大 MQCIH 版本号。

PMQCIH 操作

MQCIH 数据结构的地址。分配的存储量由 *olVersion* 指示。

原因码

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_不法版本

返回码

| 返回码 | 函数 | CompCode | 原因 | 异常终止代码 |
|-----------------------------|----------------------|--------------------------|--------------------|----------------|
| MQCRC_OK | | | | |
| MQCRC_BRIDGE_ERROR | | | MQFB_CICS | |
| MQCRC_MQ_API_ERROR | WebSphere MQ 调用名称 | WebSphere MQ CompCode | WebSphere MQ 原因 | |
| MQCRC_BRIDGE_TIMEOUT | WebSphere MQ 调用名称 | WebSphere MQ CompCode | WebSphere MQ 原因 | |
| 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 ABCODE |
| MQCRC_APPLICATION_ABEND | | | | CICS ABCODE |

ImqDeadLetterHeader C++ 类

此类封装 MQDLH 数据结构的的功能部件。

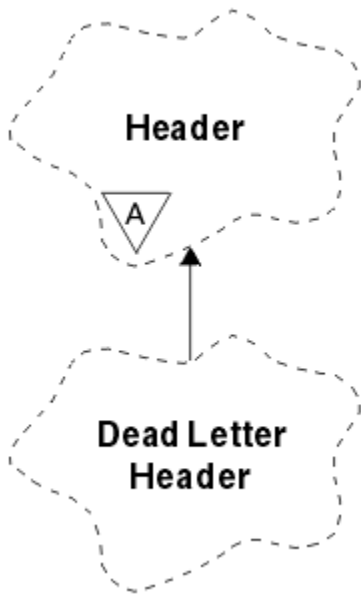


图 51: *ImqDeadLetterHeader* 类

此类的对象通常由迂到无法处理的消息的应用程序使用。包含死信头和消息内容的新消息被放置在死信队列上，并且该消息被丢弃。

- [第 1209 页的『对象属性』](#)
- [第 1210 页的『构造函数』](#)
- [第 1210 页的『重载的 *ImqItem* 方法』](#)
- [第 1210 页的『对象方法 \(公用\)』](#)
- [第 1211 页的『对象数据 \(受保护\)』](#)
- [第 1211 页的『原因码』](#)

对象属性

死信原因码

消息到达死信队列的原因。初始值为 MQRC_NONE。

目标队列管理器名称

原始目标队列管理器的名称。名称是长度为 MQ_Q_MGR_NAME_LENGTH 的字符串。其初始值为空。

目标队列名称

原始目标队列的名称。该名称是长度为 MQ_Q_NAME_LENGTH 的字符串。其初始值为空。

PUT 应用程序名称

将消息放置到死信队列的应用程序的名称。该名称是长度为 MQ_PUT_APPL_NAME_LENGTH 的字符串。其初始值为空。

PUT 应用程序类型

将消息放入死信队列的应用程序类型。初始值为零。

PUT 日期

消息放置到死信队列的日期。日期是长度为 MQ_PUT_DATE_LENGTH 的字符串。其初始值为空字符串。

PUT 时间

消息放置到死信队列的时间。时间是长度为 MQ_PUT_TIME_LENGTH 的字符串。其初始值为空字符串。

构造函数

ImqDeadLetterHeader();

缺省构造函数。

ImqDeadLetterHeader(const ImqDeadLetterHeader & 标题);

复制构造函数。

重载的 ImqItem 方法

虚拟 **ImqBoolean copyOut(ImqMessage & 消息);**

在开始时将 MQDLH 数据结构插入到消息缓冲区中，从而进一步移动现有消息数据。将 *msg format* 设置为 MQFMT_DEAD_LETTER_HEADER。

请参阅第 1216 页的『ImqHeader C++ 类』页上的 ImqHeader 类方法描述以获取更多详细信息。

虚拟 **ImqBoolean pasteIn(ImqMessage & 消息);**

从消息缓冲区读取 MQDLH 数据结构。

要成功，ImqMessage 格式必须是 MQFMT_DEAD_LETTER_HEADER。

请参阅第 1216 页的『ImqHeader C++ 类』页上的 ImqHeader 类方法描述以获取更多详细信息。

对象方法 (公用)

void 运算符 = (const ImqDeadLetterHeader & 标题);

从头复制副本实例数据，以替换现有实例数据。

MQLONG deadLetterReasonCode() const ;

返回 死信原因码。

void setDeadLetterReasonCode(const MQLONG reason);

设置 死信原因码。

ImqString destinationQueueManagerName() const ;

返回 目标队列管理器名称，并除去任何尾部空格。

void setDestinationQueueManagerName(const char * name);

设置 目标队列管理器名称。截断长于 MQ_Q_MGR_NAME_LENGTH (48 个字符) 的数据。

ImqString destinationQueue 名称() const ;

返回 目标队列名称的副本，并除去任何尾部空格。

void setDestinationQueueName(const char * name);

设置 目标队列名称。截断长度超过 MQ_Q_NAME_LENGTH (48 个字符) 的数据。

ImqString putApplication 名称() const ;

返回 put application name 的副本，除去任何尾部空格。

void setPutApplicationName(const char * name = 0);

设置 put application name。截断长度超过 MQ_PUT_APPL_NAME_LENGTH (28 个字符) 的数据。

MQLONG putApplication 类型() const ;

返回 put application type。

void setPutApplicationType(const MQLONG type = MQAT_NO_CONTEXT);

设置 put application type。

ImqString putDate() const ;

返回 put date 的副本，除去任何尾部空格。

void setPutDate(const char * date = 0);

设置 放置日期。截断长度超过 MQ_PUT_DATE_LENGTH (8 个字符) 的数据。

ImqString putTime() const ;

返回 put time 的副本，除去任何尾部空格。

void setPutTime(const char * time = 0);

设置 put time。截断长于 MQ_PUT_TIME_LENGTH (8 个字符) 的数据。

对象数据 (受保护)

MQDLH *omqdlh*

MQDLH 数据结构。

原因码

- MQRC_INCONSISTENT_FORMAT
- MQRC_STRUC_ID_ERROR
- MQRC_ENCODING_ERROR

ImqDistribution 列出 C++ 类

此类封装动态分发表，该列表引用一个或多个队列以将一条或多条消息发送到多个目标。

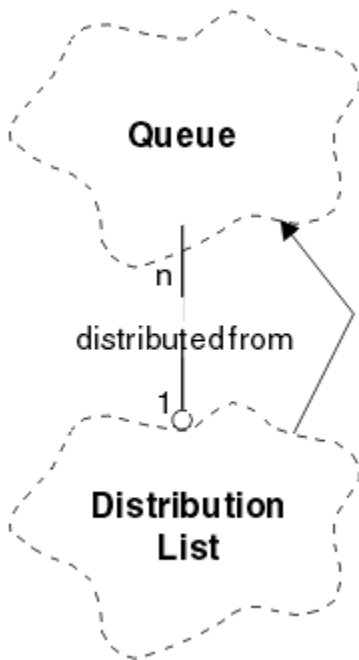


图 52: *ImqDistributionList* 类

- [第 1211 页的『对象属性』](#)
- [第 1211 页的『构造函数』](#)
- [第 1212 页的『对象方法 \(公用\)』](#)
- [第 1212 页的『对象方法 \(protected\)』](#)

对象属性

第一个分布式队列

一个或多个类对象中的第一个对象 (无特定顺序)，其中 **分发表引用** 对此对象进行寻址。

最初没有此类对象。要成功打开 *ImqDistribution* 列表，必须至少有一个此类对象。

注: 打开 *ImqDistributionList* 对象时，将自动关闭引用该对象的任何打开对象。

构造函数

ImqDistributionList ();

缺省构造函数。

ImqDistribution 列表 (const ImqDistributionList & 列表);

复制构造函数。

对象方法 (公用)

void 运算符 = (const ImqDistributionList & 列表);

在复制之前，将取消引用所有引用 **this** 对象的对象。调用此方法后，没有对象将引用 **this** 对象。

*** firstDistributed 队列() const ;**

返回 第一个分布式队列。

对象方法 (protected)

void setFirstDistributedQueue(* queue = 0);

设置 第一个分布式队列。

ImqError C++ 类

此抽象类提供有关与对象关联的错误的信息。

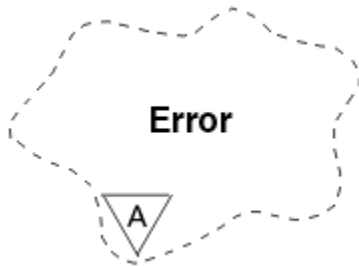


图 53: *ImqError* 类

- [第 1212 页的『对象属性』](#)
- [第 1212 页的『构造函数』](#)
- [第 1213 页的『对象方法 \(公用\)』](#)
- [第 1213 页的『对象方法 \(protected\)』](#)
- [第 1213 页的『原因码』](#)

对象属性

完成代码 (completion code)

最新的完成代码。初始值为零。可以使用以下附加值:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

原因码 (reason code)

最新原因码。初始值为零。

构造函数

ImqError();

缺省构造函数。

ImqError(const ImqError & 错误);

复制构造函数。

对象方法 (公用)

void 运算符 = (const ImqError & 错误);
从 *error* 复制实例数据, 替换现有实例数据。

void clearErrorCodes();
将完成代码和原因码都设置为零。

MQLONG completionCode() const ;
返回完成代码。

MQLONG reasonCode() const ;
返回原因码。

对象方法 (protected)

ImqBoolean checkReadPointer(const void * pointer, const size_t length);
验证指针和长度的组合是否对只读访问有效, 如果成功, 将返回 TRUE。

ImqBoolean checkWritePointer(const void * 指针, const size_t length);
验证指针和长度的组合是否对读写访问有效, 如果成功, 将返回 TRUE。

void setCompletion 代码(const MQLONG 代码 = 0);
设置完成代码。

void setReasonCode(const MQLONG code = 0);
设置原因码。

原因码

- MQRC_BUFFER_ERROR

ImqGetMessageOptions C++ 类

此类封装 MQGMO 数据结构

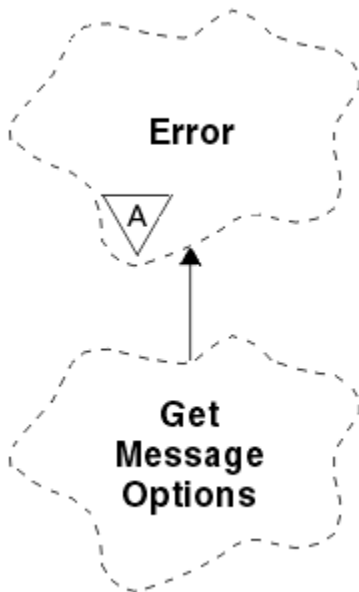


图 54: *ImqGetMessageOptions* 类

- [第 1214 页的『对象属性』](#)
- [第 1215 页的『构造函数』](#)
- [第 1215 页的『对象方法 \(公用\)』](#)
- [第 1216 页的『对象方法 \(protected\)』](#)

- [第 1216 页的『对象数据 \(受保护\)』](#)
- [第 1216 页的『原因码』](#)

对象属性

集团状态

一组消息的消息状态。初始值为 MQGS_NOT_IN_GROUP。可以使用以下附加值:

- MQGS_MSG_IN_GROUP
- MQGS_LAST_MSG_IN_GROUP

匹配选项

用于选择入局消息的选项。初始值为 MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID。可以使用以下附加值:

- MQMO_GROUP_ID
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN
- MQMO_NONE

消息标记 (message token)

消息令牌。长度为 MQ_MSG_TOKEN_LENGTH 的二进制值 (MQBYTE16)。初始值为 MQMTOK_NONE。

选项

适用于消息的选项。初始值为 MQGMO_NO_WAIT。可以使用以下附加值:

- 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

已解析队列名称

已解析的队列名称。此属性是只读的。名称的长度不得超过 48 个字符，并且可以使用空值来填充该长度。初始值为空字符串。

返回的长度

返回的长度。初始值为 MQRL_UNDEFINED。此属性是只读的。

分段

对消息进行分段的能力。初始值为 MQSEG_禁止。可以使用附加值 MQSEG_ALLOWED。

段状态

消息的分段状态。初始值为 MQSS_NOT_A_SEGMENT。可以使用以下附加值:

- MQSS_SEGMENT
- MQSS_LAST_SEGMENT

同步点参与

在同步点控制下检索消息时为 TRUE。

等待时间间隔

类 **get** 方法在等待合适的消息到达时暂停的时间长度 (如果还没有可用的消息)。初始值为零, 这将影响无限期等待。可以使用附加值 MQWI_UNLIMITED。除非 **选项** 包含 MQGMO_WAIT, 否则将忽略此属性。

构造函数

ImqGetMessageOptions();

缺省构造函数。

ImqGetMessageOptions(const ImqGetMessageOptions & 格莫);

复制构造函数。

对象方法 (公用)

void 运算符 = (const ImqGetMessageOptions & 格莫);

从 *gmo* 复制实例数据, 替换现有实例数据。

MQCHAR groupStatus() const ;

返回 组状态。

void setGroupStatus(const MQCHAR status);

设置 组状态。

MQLONG matchOptions() const ;

返回 匹配选项。

void setMatchOptions(const MQLONG options);

设置 匹配选项。

ImqBinary messageToken() 康斯特;

返回 消息令牌。

ImqBoolean setMessageToken(const ImqBinary & 令牌);

设置 消息令牌。 *token* 的 数据长度 必须为零或 MQ_MSG_TOKEN_LENGTH。如果成功, 此方法将返回 TRUE。

void setMessageToken (const MQBYTE16 token = 0);

设置 消息令牌。 *token* 可以为零, 这与指定 MQMTOK_NONE 相同。如果 *token* 非零, 那么它必须处理二进制数据的 MQ_MSG_TOKEN_LENGTH 字节。

使用预定义值 (例如 MQMTOK_NONE) 时, 可能不需要进行强制类型转换以确保签名匹配, 例如 (MQBYTE *) MQMTOK_NONE。

MQLONG 选项() const ;

返回 选项。

void setOptions(const MQLONG 选项);

设置 选项, 包括 **syncpoint** 参与 值。

ImqString resolvedQueue 名称() const ;

返回 已解析队列名称的副本。

MQLONG returnedLength() 康斯特;

返回 返回的长度。

MQCHAR 分段() const ;

返回 分段。

void setSegmentation(const MQCHAR value);

设置 分段。

MQCHAR segmentStatus() const ;

返回 段状态。

void setSegmentStatus(const MQCHAR status);

设置 段状态。

ImqBoolean syncPoint 参与() const ;

返回 **syncpoint** 参与 值, 如果 **选项** 包含 MQGMO_SYNCPOINT 或 MQGMO_SYNCPOINT_IF_PERSISTENT, 那么此值为 TRUE。

void setSyncPointParticipation(const ImqBoolean sync);

设置 **syncpoint** 参与 值。如果 *sync* 为 TRUE, 请更改 **选项** 以包含 MQGMO_SYNCPOINT, 并同时排除 MQGMO_NO_SYNCPOINT 和 MQGMO_SYNCPOINT_IF_PERSISTENT。如果 *sync* 为 FALSE, 请更改 **选项** 以包含 MQGMO_NO_SYNCPOINT, 并同时排除 MQGMO_SYNCPOINT 和 MQGMO_SYNCPOINT_IF_PERSISTENT。

MQLONG waitInterval() const ;

返回 等待时间间隔。

void setWaitInterval(const MQLONG interval);

设置 等待时间间隔。

对象方法 (protected)

静态 **void setVersionSupported(const MQLONG);**

设置 MQGMO 版本。缺省为 MQGMO_VERSION_3。

对象数据 (受保护)

MQGMO omqgmo

MQGMO 版本 2 数据结构。仅访问 MQGMO_VERSION_2 支持的 MQGMO 字段。

PMQGMO opgmo

MQGMO 数据结构的地址。此地址的版本号在 *olVersion* 中指示。在访问 MQGMO 字段之前检查版本号, 以确保存在这些字段。

MQLONG olVersion

由 *opgmo* 寻址的 MQGMO 数据结构的版本号。

原因码

- MQRC_BINARY_DATA_LENGTH_ERROR

ImqHeader C++ 类

此抽象类封装 MQDLH 数据结构的公共功能。

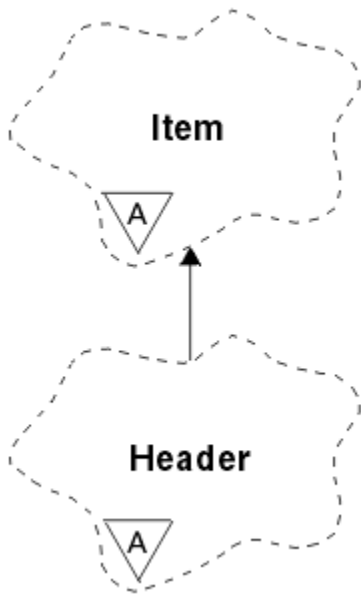


图 55: *ImqHeader* 类

- [第 1217 页的『对象属性』](#)
- [第 1217 页的『构造函数』](#)
- [第 1217 页的『对象方法 \(公用\)』](#)

对象属性

字符集

原始编码字符集标识。初始 MQCCSI_Q_MGR。

编码

原始编码。初始 MQENC_NATIVE。

Format

原始格式。初始 MQFMT_NONE。

头标志

初始值为:

- 对于 *ImqDeadLetterHeader* 类的对象为零
- 针对 *ImqIMSBridge* 头类的对象的 MQIIH_NONE
- 针对 *ImqReference* 头类的对象的 MQRMHF_LAST
- 针对 *ImqCICSBridge* 头类的对象的 MQCIH_NONE
- 针对 *ImqWork* 头类的对象的 MQWIH_NONE

构造函数

ImqHeader();

缺省构造函数。

ImqHeader(const ImqHeader & 标题);

复制构造函数。

对象方法 (公用)

void 运算符 = (const ImqHeader & 标题);

从头复制实例数据，替换现有实例数据。

虚拟 **MQLONG** `characterSet()` `const` ;

返回 字符集。

virtual void `setCharacterSet(const MQLONG ccsid = MQCCSI_Q_MGR);`

设置 字符集。

虚拟 **MQLONG** `encoding()` `const` ;

返回 `encoding`。

虚拟空 `setEncoding(const MQLONG encoding = MQENC_NATIVE);`

设置 `encoding`。

虚拟 **ImqString** `format()` `const` ;

返回 `format` 的副本，包括尾部空格。

虚拟空 `setFormat(const char * name = 0);`

设置 `format`，填充为带有尾部空格的 8 字符。

虚拟 **MQLONG** `headerFlags()` `const` ;

返回 头标志。

virtual void `setHeaderFlags(const MQLONG flags = 0);`

设置 头标志。

ImqIMSBridge 头 C++ 类

此类封装 MQIIH 数据结构的功能。

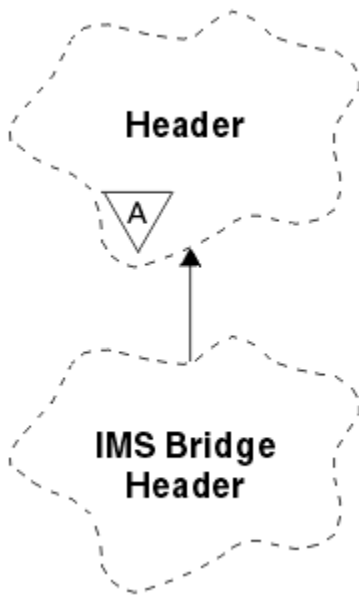


图 56: *ImqIMSBridge* 头类

此类的对象由通过 WebSphere MQ for z/OS 向 IMS 网桥发送消息的应用程序使用。

注: `ImqHeader` 字符集 和 `encoding` 必须具有缺省值，并且不得设置为任何其他值。

- [第 1219 页的『对象属性』](#)
- [第 1219 页的『构造函数』](#)
- [第 1219 页的『重载的 `ImqItem` 方法』](#)
- [第 1219 页的『对象方法 \(公用\)』](#)
- [第 1220 页的『对象数据 \(受保护\)』](#)
- [第 1220 页的『原因码』](#)

对象属性

鉴别符 (authenticator)

RACF 密码或通行票，长度为 MQ_AUTHENTICATOR_LENGTH。初始值为 MQIAUT_NONE。

落实方式

落实方式。请参阅 OTMA 用户指南，以获取有关 IMS 落实方式的更多信息。初始值为 MQICM_COMMIT_THEN_SEND。可以使用附加值 MQICM_SEND_THEN_COMMIT。

逻辑终端覆盖

长度为 MQ_LTERM_OVERRIDE_LENGTH 的逻辑终端覆盖。初始值为空字符串。

消息格式化服务映射名称

MFS 映射名称，长度为 MQ_MFS_MAP_NAME_LENGTH。初始值为空字符串。

回复格式

长度为 MQ_FORMAT_LENGTH 的任何应答的格式。初始值为 MQFMT_NONE。

安全范围

IMS 安全性处理的作用域。初始值为 MQISS_CHECK。可以使用附加值 MQISS_FULL。

事务实例标识

事务实例身份，长度为 MQ_TRAN_INSTANCE_ID_LENGTH 的二进制 (MQBYTE16) 值。初始值为 MQITII_NONE。

事务状态

IMS 对话的状态。初始值为 MQITS_NOT_IN_CONVERSATION。可以使用附加值 MQITS_IN_CONVERSATION。

构造函数

ImqIMSBridgeHeader ();

缺省构造函数。

ImqIMSBridge 头 (const ImqIMSBridge 头和头);

复制构造函数。

重载的 ImqItem 方法

虚拟 ImqBoolean copyOut(ImqMessage & 消息);

在开始时将 MQIIH 数据结构插入到消息缓冲区中，从而进一步移动现有消息数据。将 *msg* 格式设置为 MQFMT_IMS。

请参阅父类方法描述以获取更多详细信息。

虚拟 ImqBoolean pasteIn(ImqMessage & 消息);

从消息缓冲区读取 MQIIH 数据结构。

要成功，*msg* 对象的 **encoding** 必须是 MQENC_NATIVE。使用 MQGMO_CONVERT 检索到 MQENC_NATIVE 的消息。

要成功，*ImqMessage* 格式必须是 MQFMT_IMS。

请参阅父类方法描述以获取更多详细信息。

对象方法 (公用)

void 运算符 = (const ImqIMSBridgeHeader & 标题);

从头复制实例数据，替换现有实例数据。

ImqString authenticator() const ;

返回 **authenticator** 的副本，以尾部空格填充至长度 MQ_AUTHENTICATOR_LENGTH。

void setAuthenticator(const char * name);

设置 **鉴别符**。

MQCHAR commitMode() const ;

返回 **落实方式**。

void setCommitMode(const MQCHAR mode);

设置 落实方式。

ImqString logicalTerminal 覆盖() const ;

返回 逻辑终端覆盖的副本。

void setLogicalTerminalOverride(const char * override);

设置 逻辑终端覆盖。

ImqString messageFormatServicesMapName() const ;

返回 消息格式服务映射名称的副本。

void setMessageFormatServicesMapName(const char * name);

设置 消息格式服务映射名称。

ImqString replyTo 格式() const ;

返回 应答格式的副本，以尾部空格填充，长度为 MQ_FORMAT_LENGTH。

void setReplyToFormat(const char * format);

设置 应答格式，用尾部空格填充到长度 MQ_FORMAT_LENGTH。

MQCHAR securityScope() const ;

返回 安全作用域。

void setSecurityScope(const MQCHAR scope);

设置 安全作用域。

ImqBinary transactionInstanceId() const ;

返回 事务实例标识的副本。

ImqBoolean setTransactionInstanceId(const ImqBinary & 标识);

设置 事务实例标识。 *token* 的 数据长度 必须为零或 MQ_TRAN_INSTANCE_ID_LENGTH。 如果成功，此方法将返回 TRUE。

void setTransactionInstanceId(const MQBYTE16 id = 0);

设置 事务实例标识。 *id* 可以为零，这与指定 MQITII_NONE 相同。 如果 *id* 非零，那么它必须寻址二进制数据的 MQ_TRAN_INSTANCE_ID_LENGTH 字节。 使用预定义值 (例如 MQITII_NONE) 时， 可能需要进行强制类型转换以确保签名匹配，例如 (MQBYTE *) MQITII_NONE。

MQCHAR transactionState() const ;

返回 事务状态。

void setTransactionState(const MQCHAR state);

设置 事务状态。

对象数据 (受保护)

MQIIH omqiih

MQIIH 数据结构。

原因码

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

ImqItem C++ 类

此抽象类表示消息中的一项 (可能是若干项中的一项)。

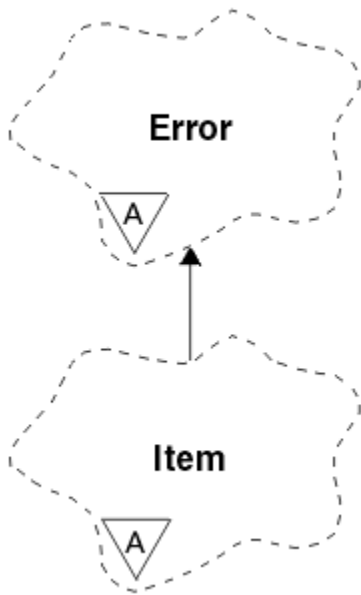


图 57: *ImqItem* 类

项在消息缓冲区中并置在一起。每个特殊化都与以结构标识开头的特定数据结构相关联。

此抽象类中的多态方法允许将项复制到消息或从消息复制项。*ImqMessage* 类 **readItem** 和 **writeItem** 方法提供了另一种调用这些多态方法的样式，这些多态方法对于应用程序更自然。

- [第 1221 页的『对象属性』](#)
- [第 1221 页的『构造函数』](#)
- [第 1221 页的『类方法 \(public\)』](#)
- [第 1221 页的『对象方法 \(公用\)』](#)
- [第 1222 页的『原因码』](#)

对象属性

结构标识

数据结构开头的由四个字符组成的字符串。此属性是只读的。对于派生类，请考虑此属性。不会自动包含此内容。

构造函数

ImqItem();

缺省构造函数。

ImqItem(const ImqItem & 项);

复制构造函数。

类方法 (public)

静态 ImqBoolean structureId 是(**const 字符 * 要测试的结构标识, const ImqMessage & 消息**);

如果入局 *msg* 中下一个 *ImqItem* 的 **结构标识** 与 *structure-id-to-test* 相同，那么返回 TRUE。下一项标识为当前由 *ImqCache* **数据指针** 寻址的消息缓冲区的该部分。此方法依赖于 **结构标识**，因此不保证适用于所有 *ImqItem* 派生类。

对象方法 (公用)

void 运算符 = (const ImqItem & 项);

从项复制实例数据，替换现有实例数据。

虚拟 **ImqBoolean copyOut(ImqMessage & 消息) = 0;**

将此对象作为下一个项写入出局消息缓冲区，并将其附加到任何现有项。如果写操作成功，请增大 ImqCache **数据长度**。如果成功，此方法将返回 TRUE。

覆盖此方法以使用特定子类。

虚拟 **ImqBoolean pasteIn(ImqMessage & 消息) = 0;**

从入局消息缓冲区 破坏性地 读取此对象。读取具有破坏性，因为将 ImqCache **数据指针** 移开。但是，缓冲区内容保持不变，因此可以通过重置 ImqCache **数据指针** 来重新读取数据。

此对象的 (子) 类必须与 *msg* 对象的消息缓冲区中下一个找到的 **structure id** 一致。

msg 对象的 **encoding** 应该是 MQENC_NATIVE。建议使用设置为 MQENC_NATIVE 的 ImqMessage **encoding** 以及包含 MQGMO_CONVERT 的 ImqGetMessageOptions **选项** 来检索消息。

如果读操作成功，那么会减小 ImqCache **数据长度**。如果成功，此方法将返回 TRUE。

覆盖此方法以使用特定子类。

原因码

- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_IN 区内的缓冲区
- MQRC_IN 有数据

ImqMessage C++ 类

此类封装 MQMD 数据结构，还处理消息数据的构造和重建。

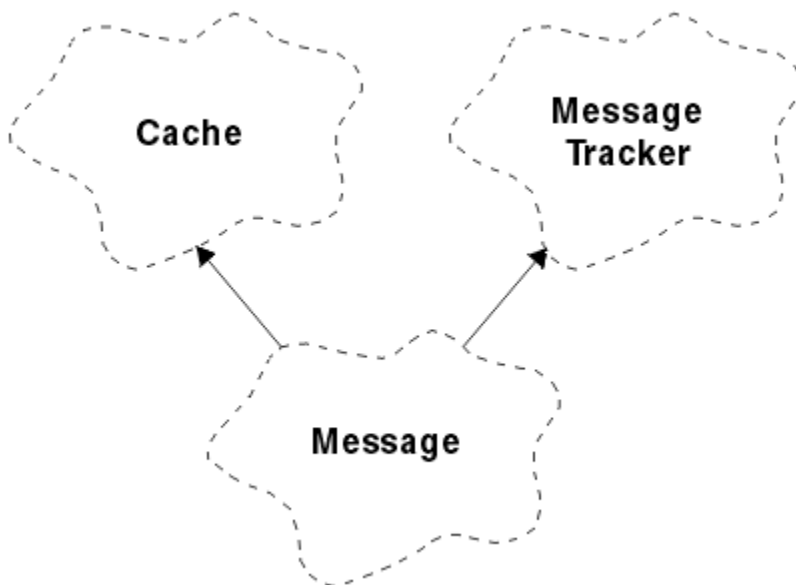


图 58: ImqMessage 类

- [第 1223 页的『对象属性』](#)
- [第 1226 页的『构造函数』](#)
- [第 1226 页的『对象方法 \(公用\)』](#)
- [第 1228 页的『对象方法 \(protected\)』](#)
- [第 1228 页的『对象数据 \(受保护\)』](#)

对象属性

应用程序标识数据

与消息关联的身份信息。初始值为空字符串。

应用程序源数据

与消息关联的源信息。初始值为空字符串。

回退计数

暂时检索并随后回退消息的次数。初始值为零。此属性是只读的。

字符集

编码字符集标识。初始值为 MQCCSI_Q_MGR。可以使用以下附加值：

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

您还可以使用您选择的编码字符集标识。有关这方面的信息，请参阅第 818 页的『[代码页转换](#)』。

编码

消息数据的机器编码。初始值为 MQENC_NATIVE。

到期

时间相关的数量，用于控制 WebSphere MQ 在废弃消息之前保留未检索消息的时间长度。初始值为 MQEI_UNLIMITED。

Format

用于描述缓冲区中数据布局的格式 (模板) 的名称。长度超过 8 个字符的名称将截断为 8 个字符。名称总是用空格填充为 8 个字符。初始常量值为 MQFMT_NONE。可以使用以下其他常量：

- 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

您还可以使用您选择的特定于应用程序的字符串。有关此操作的更多信息，请参阅消息描述符 (MQMD) 的第 377 页的『[Format \(MQCHAR8\)](#)』字段。

消息标志

分段控制信息。初始值为 MQMF_SEGMENTATION_ALLOWED。可以使用以下附加值：

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT

- MQMF_LAST_SEGMENT
- MQMF_NONE

消息类型

消息的广义分类。初始值为 MQMT_DATAGRAM。可以使用以下附加值:

- MQMT_SYSTEM_FIRST
- MQMT_SYSTEM_LAST
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- MQMT_REPORT
- MQMT_APPL_FIRST
- MQMT_APPL_LAST

您还可以使用您选择的特定于应用程序的值。有关此操作的更多信息, 请参阅消息描述符 (MQMD) 的 [第 385 页的『MsgType \(MQLONG\)』](#) 字段。

偏移量 (offset)

偏移信息。初始值为零。

原始长度

分段消息的原始长度。初始值为 MQOL_UNDEFINED。

持久性

指示消息很重要, 必须始终使用持久存储器进行备份。此选项意味着性能下降。初始值为 MQPER_PERSISTENCE_AS_Q_DEF。可以使用以下附加值:

- MQPER_PERSISTENT
- MQPER_NOT_PERSISTENT

priority

传输和交付的相对优先级。具有相同优先级的消息通常按提供这些消息的顺序进行传递 (尽管必须满足若干条件才能保证这一点)。初始值为 MQPRI_PRIORITY_AS_Q_DEF。

属性验证

指定在设置消息的属性时是否应进行属性验证。初始值为 MQCMHO_DEFAULT_VALIDATION。可以使用以下附加值:

- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

以下方法用于 **属性验证**:

MQLONG propertyValidation() 康斯特;

返回 **属性验证** 选项。

void setPropertyValidation (const MQLONG 选项);

设置 **属性验证** 选项。

PUT 应用程序名称

放置消息的应用程序的名称。初始值为空字符串。

PUT 应用程序类型

放置消息的应用程序的类型。初始值为 MQAT_NO_CONTEXT。可以使用以下附加值:

- 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

您还可以使用您选择的特定于应用程序的字符串。有关此操作的更多信息，请参阅消息描述符 (MQMD) 的第 390 页的『PutApplType (MQLONG)』字段。

PUT 日期

放入消息的日期。初始值为空字符串。

PUT 时间

放入消息的时间。初始值为空字符串。

应答队列管理器名称

应该向其发送任何应答的队列管理器的名称。初始值为空字符串。

应答队列名称

应将任何应答发送到的队列的名称。初始值为空字符串。

report

与消息关联的反馈信息。初始值为 MQRO_NONE。可以使用以下附加值：

- 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

其中 * 指示 WebSphere MQ for z/OS 上不支持的值。

序号 (sequence number)

标识消息在组中的顺序的信息。初始值为 1。

消息总长度

最近一次尝试读取消息期间可用的字节数。如果最后一条消息被截断，或者如果由于发生截断而未读取最后一条消息，那么此数字将大于 `ImqCache 消息长度`。此属性是只读的。初始值为零。

此属性在涉及截断消息的任何情况下都很有用。

用户标识

与消息关联的用户身份。初始值为空字符串。

构造函数

ImqMessage();

缺省构造函数。

ImqMessage(const ImqMessage & 消息);

复制构造函数。请参阅 `operator =` 方法以获取详细信息。

对象方法 (公用)

void 运算符 = (const ImqMessage & 消息);

从 `msg` 复制 MQMD 和消息数据。如果用户为此对象提供了缓冲区，那么复制的数据量将限制为可用缓冲区大小。否则，系统将确保具有足够大小的缓冲区可用于复制的数据。

ImqString applicationIdData() const ;

返回 应用程序标识数据的副本。

void setApplicationIdData(const char * data = 0);

设置 应用程序标识数据。

ImqString applicationOriginData() const ;

返回 应用程序原始数据的副本。

void setApplicationOriginData(const char * data = 0);

设置 应用程序源数据。

MQLONG backoutCount() const ;

返回 回退计数。

MQLONG characterSet() const ;

返回 字符集。

void setCharacterSet(const MQLONG ccsid = MQCCSI_Q_MGR);

设置 字符集。

MQLONG 编码() const ;

返回 `encoding`。

void setEncoding(const MQLONG encoding = MQENC_NATIVE);

设置 `encoding`。

MQLONG 到期() const ;

返回 到期。

void setExpiry(const MQLONG 到期);

设置 到期。

ImqString 格式() const ;

返回 `format` 的副本，包括尾部空格。

ImqBoolean formatIs(const char * *format-to-test*) const ;
 如果 **format** 与 *format-to-test* 相同, 那么返回 TRUE。

void setFormat(const char * *name* = 0);
 设置 **format**, 填充为带有尾部空格的 8 个字符。

MQLONG messageFlags() const ;
 返回 消息标志。

void setMessageFlags(const MQLONG *flags*);
 设置 消息标志。

MQLONG messageType() const ;
 返回 消息类型。

void setMessageType(const MQLONG *type*);
 设置 消息类型。

MQLONG offset() const ;
 返回 **offset**。

void setOffset(const MQLONG *offset*);
 设置 **offset**。

MQLONG originalLength() const ;
 返回 原始长度。

void setOriginalLength(const MQLONG *length*);
 设置 原始长度。

MQLONG 持久性() const ;
 返回 持久性。

void setPersistence(const MQLONG 持久性);
 设置 持久性。

MQLONG 优先级() const ;
 返回 **priority**。

void setPriority(const MQLONG *priority*);
 设置 优先级。

ImqString putApplication 名称() const ;
 返回 **put application name** 的副本。

void setPutApplicationName(const char * *name* = 0);
 设置 **put application name**。

MQLONG putApplication 类型() const ;
 返回 **put application type**。

void setPutApplicationType(const MQLONG *type* = MQAT_NO_CONTEXT);
 设置 **put application type**。

ImqString putDate() const ;
 返回 **put date** 的副本。

void setPutDate(const char * *date* = 0);
 设置 放置日期。

ImqString putTime() const ;
 返回 **put time** 的副本。

void setPutTime(const char * *time* = 0);
 设置 **put time**。

ImqBoolean readItem(ImqItem & 项);
 使用 **ImqItem pasteIn** 方法从消息缓冲区读取 *item* 对象。 如果成功, 将返回 TRUE。

ImqString replyToQueueManager 名称() const ;
 返回 应答队列管理器名称的副本。

void setReplyToQueueManagerName(const char * *name* = 0);
 设置 应答队列管理器名称。

ImqString replyToQueueName() const ;

返回 应答队列名称的副本。

void setReplyToQueueName(const char * name = 0);

设置 应答队列名称。

MQLONG report() const ;

返回 report。

void setReport(const MQLONG report);

设置 报告。

MQLONG sequenceNumber() const ;

返回 序号。

void setSequenceNumber(const MQLONG number);

设置 序号。

size_t totalMessageLength() const ;

返回 总消息长度。

ImqString userId() const ;

返回 用户标识的副本。

void setUserId(const char * id = 0);

设置 用户标识。

ImqBoolean writeItem(ImqItem & 项);

使用 ImqItem **copyOut** 方法从 *item* 对象写入消息缓冲区。编写可以采用插入, 替换或附加的形式: 这取决于 *item* 对象的类。如果成功, 此方法将返回 TRUE。

对象方法 (protected)

静态 **void setVersionSupported(const MQLONG);**

设置 MQMD 版本。缺省为 MQMD_VERSION_2。

对象数据 (受保护)

MQMD1 omqmd

(仅限 WebSphere MQ for z/OS。)MQMD 数据结构。

MQMD2 omqmd

(除 z/OS 以外的平台。)MQMD 数据结构。

ImqMessageTracker C++ 类

此类封装了可与任一对象关联的 ImqMessage 或 ImqQueue 对象的那些属性。

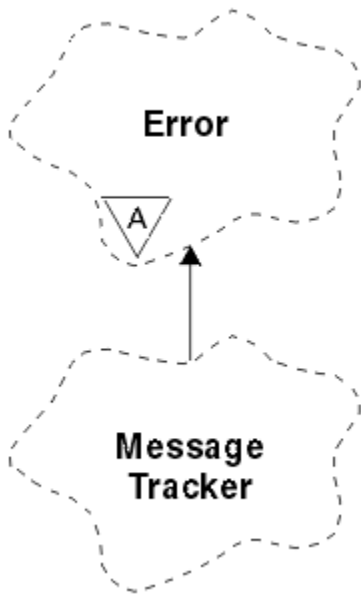


图 59: *ImqMessage* 跟踪程序类

此类与第 1183 页的『[ImqMessage 跟踪程序交叉引用](#)』中列出的 MQI 调用相关。

- [第 1229 页的『对象属性』](#)
- [第 1230 页的『构造函数』](#)
- [第 1230 页的『对象方法 \(公用\)』](#)
- [第 1231 页的『原因码』](#)

对象属性

记帐标记

长度为 MQ_ACCOUNTING_TOKEN_LENGTH 的二进制值 (MQBYTE32)。初始值为 MQACT_NONE。

相关标识

指定用于关联消息的长度为 MQ_CORREL_ID_LENGTH 的二进制值 (MQBYTE24)。初始值为 MQCI_NONE。可以使用附加值 MQCI_NEW_SESSION。

Feedback

要随消息一起发送的反馈信息。初始值为 MQFB_NONE。可以使用以下附加值:

- 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_负数
- MQFB_DATA_LENGTH_TOO_BIG
- MQFB_BUFFER_OVERFLOW

- MQFB_LENGTH_OFF_BY_ONE
- MQFB_IIH_ERROR
- MQFB_NOT_AUTHORIZED_FOR_IM
- MQFB_IMS_ERROR
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_AB 已结束
- 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

您还可以使用您选择的特定于应用程序的字符串。有关此操作的更多信息，请参阅消息描述符 (MQMD) 的第 373 页的『[Feedback \(MQLONG\)](#)』字段。

组标识

队列中唯一的长度为 MQ_GROUP_ID_LENGTH 的二进制值 (MQBYTE24)。初始值为 MQGI_NONE。

消息标识

队列中唯一的长度为 MQ_MSG_ID_LENGTH 的二进制值 (MQBYTE24)。初始值为 MQMI_NONE。

构造函数

ImqMessageTracker ();

缺省构造函数。

ImqMessage 跟踪程序 (const ImqMessageTracker & 跟踪程序);

复制构造函数。请参阅 **operator =** 方法以获取详细信息。

对象方法 (公用)

void 运算符 = (const ImqMessageTracker & 跟踪程序);

从跟踪程序复制实例数据，替换现有实例数据。

ImqBinary accountingToken() const ;

返回 记帐令牌的副本。

ImqBoolean setAccounting 令牌(const ImqBinary & 令牌);

设置 记帐令牌。令牌的数据长度必须为零或 MQ_ACCOUNTING_TOKEN_LENGTH。如果成功，此方法将返回 TRUE。

void setAccountingToken(const MQBYTE32 token = 0);

设置 记帐令牌。token 可以为零，这与指定 MQACT_NONE 相同。如果 token 非零，那么它必须寻址二进制数据的 MQ_ACCOUNTING_TOKEN_LENGTH 字节。使用预定义值 (例如 MQACT_NONE) 时，可能需要进行强制类型转换以确保签名匹配; 例如 (MQBYTE *) MQACT_NONE。

ImqBinary correlationId() const ;

返回 相关标识的副本。

ImqBoolean setCorrelation 标识(const ImqBinary & 令牌);

设置 相关标识。 *token* 的 数据长度 必须为零或 MQ_CORREL_ID_LENGTH。 如果成功， 此方法将返回 TRUE。

void setCorrelationId(const MQBYTE24 id = 0);

设置 相关标识。 *id* 可以为零， 这与指定 MQCI_NONE 相同。 如果 *id* 非零， 那么它必须寻址二进制数据的 MQ_CORREL_ID_LENGTH 字节。 使用预定义值 (例如 MQCI_NONE) 时， 可能需要进行强制类型转换以确保签名匹配; 例如 (MQBYTE *) MQCI_NONE。

MQLONG 反馈() const ;

返回 反馈。

void setFeedback(const MQLONG feedback);

设置 反馈。

ImqBinary groupId() const ;

返回 组标识的副本。

ImqBoolean setGroup 标识(const ImqBinary & 令牌);

设置 组标识。 *token* 的 数据长度 必须为零或 MQ_GROUP_ID_LENGTH。 如果成功， 此方法将返回 TRUE。

void setGroupId(const MQBYTE24 id = 0);

设置 组标识。 *id* 可以为零， 这与指定 MQGI_NONE 相同。 如果 *id* 非零， 那么它必须处理二进制数据的 MQ_GROUP_ID_LENGTH 字节。 使用预定义值 (例如 MQGI_NONE) 时， 您可能需要进行强制类型转换以确保签名匹配， 例如 (MQBYTE *) MQGI_NONE。

ImqBinary messageId() const ;

返回 消息标识的副本。

ImqBoolean setMessage 标识(const ImqBinary & 令牌);

设置 消息标识。 令牌的 数据长度 必须为零或 MQ_MSG_ID_LENGTH。 如果成功， 此方法将返回 TRUE。

void setMessageId(const MQBYTE24 id = 0);

设置 消息标识。 *id* 可以为零， 这与指定 MQMI_NONE 相同。 如果 *id* 非零， 那么它必须处理二进制数据的 MQ_MSG_ID_LENGTH 字节。 使用预定义值 (例如 MQMI_NONE) 时， 可能需要进行强制类型转换以确保签名匹配， 例如 (MQBYTE *) MQMI_NONE。

原因码

- MQRC_BINARY_DATA_LENGTH_ERROR

ImqNamelist C++ 类

此类封装名称列表。

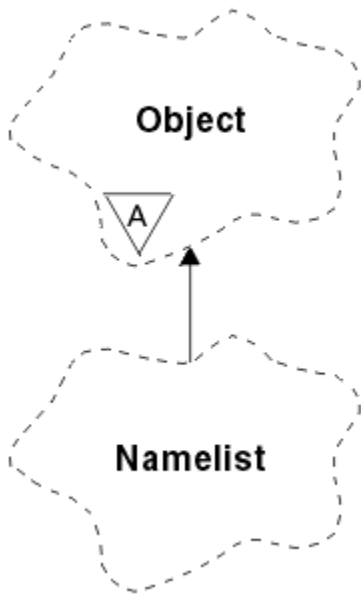


图 60: *ImqNamelist* 类

此类与第 1183 页的『[ImqNamelist 交叉引用](#)』中列出的 MQI 调用相关。

- [第 1232 页的『对象属性』](#)
- [第 1232 页的『构造函数』](#)
- [第 1232 页的『对象方法 \(公用\)』](#)
- [第 1233 页的『原因码』](#)

对象属性

名称计数

namelist names 中的对象名数。此属性是只读的。

名称列表名称

对象名，其编号由 **name count** 指示。此属性是只读的。

构造函数

ImqNamelist();

缺省构造函数。

ImqNamelist(const ImqNamelist & 列表);

复制构造函数。ImqObject 打开状态为 false。

ImqNamelist(const char * name);

将 ImqObject 名称设置为 **name**。

对象方法 (公用)

void operator = (const ImqNamelist & 列表);

从 *list* 复制实例数据，替换现有实例数据。ImqObject 打开状态为 false。

ImqBoolean nameCount(MQLONG & 计数);

提供 **name count** 的副本。如果成功，将返回 TRUE。

MQLONG nameCount ();

返回 **name count**，而不指示任何可能的错误。

ImqBoolean namelistName (const MQLONG 索引, ImqString 和 名);

通过基于零的索引提供一个 **名称列表名称** 的副本。如果成功，将返回 TRUE。

ImqString namelistName (const MQLONG index);

通过基于零的索引返回其中一个 **名称列表名称**，而不指示任何可能的错误。

原因码

- MQRC_INDEX_ERROR
- MQRC_INDEX_NOT_PRESENT

ImqObject C++ 类

这个类是抽象的。销毁此类的对象时，会自动将其关闭，并断开其 ImqQueueManager 连接。

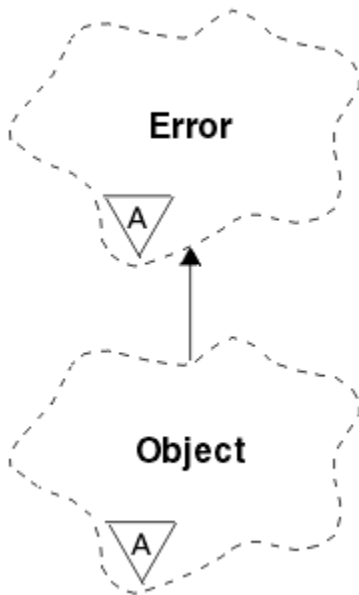


图 61: ImqObject 类

此类与第 1184 页的『ImqObject 交叉引用』中列出的 MQI 调用相关。

- [第 1233 页的『类属性』](#)
- [第 1234 页的『对象属性』](#)
- [第 1235 页的『构造函数』](#)
- [第 1235 页的『类方法 \(public\)』](#)
- [第 1235 页的『对象方法 \(公用\)』](#)
- [第 1237 页的『对象方法 \(protected\)』](#)
- [第 1237 页的『对象数据 \(受保护\)』](#)
- [第 1237 页的『原因码』](#)
-

类属性

行为 (behavior)

控制隐式打开的行为。

IMQ_IMPL_OPEN (8L)

允许隐式打开。这是缺省值。

对象属性

变更日期

变更日期。此属性是只读的。

变更时间

变更时间。此属性是只读的。

备用用户标识

备用用户标识，最多为 MQ_USER_ID_LENGTH 字符。初始值为空字符串。

备用安全标识

备用安全标识。长度为 MQ_SECURITY_ID_LENGTH 的二进制值 (MQBYTE40)。初始值为 MQSID_NONE。

关闭选项

关闭对象时应用的选项。初始值是 MQCO_NONE。在隐式重新打开操作期间将忽略此属性，其中始终使用值 MQCO_NONE。

连接参考

对 ImqQueueManager 对象的引用，该对象提供与 (本地) 队列管理器的必需连接。对于 ImqQueueManager 对象，它是对象本身。初始值为零。

注: 请勿将此与用于标识指定队列的队列管理器 (可能是远程队列) 的 **队列管理器名称** 混淆。

description

队列管理器，队列，名称列表或进程的描述性名称 (最多 64 个字符)。此属性是只读的。

name

队列管理器，队列，名称列表或进程的名称 (最多 48 个字符)。初始值为空字符串。在 **打开** 之后，模型队列的名称将更改为生成的动态队列的名称。

注: ImqQueueManager 可以具有空名称，表示缺省队列管理器。成功 **打开** 后，名称将更改为实际队列管理器。ImqDistribution 列表是动态的，必须具有空名称。

下一个受管对象

这是此类的下一个对象，没有特殊顺序，具有与此对象相同的 **连接引用**。初始值为零。

打开选项

打开对象时适用的选项。初始值为 MQOO_INQUIRE。有两种方法可以设置相应的值:

1. 请勿设置 **open options**，也不要使用 **open** 方法。WebSphere MQ 会自动调整 **打开选项**，并根据需要自动打开，重新打开和关闭对象。这可能会导致不必要的重新打开操作，因为 WebSphere MQ 使用 **openFor** 方法，并且仅以递增方式添加 **打开选项**。
2. 在使用导致 MQI 调用的任何方法之前设置 **open options** (请参阅第 1178 页的『C++ 和 MQI 交叉引用』)。这可确保不会发生不必要的重新打开操作。如果可能发生任何潜在的重新打开问题，请显式设置打开选项 (请参阅 [重新打开](#))。

如果使用 **open** 方法，那么 **必须** 确保 **open options** 首先适用。但是，使用 **open** 方法不是必需的; WebSphere MQ 仍然表现出与案例 1 相同的行为，但在这种情况下，行为是有效的。

零不是有效值; 请在尝试打开对象之前设置相应的值。可以使用 **setOpen 选项** (*lOpen 选项*) 来完成此操作后跟 **open()** 或 **openFor (lRequiredOpenOption)**。

注:

1. MQOO_OUTPUT 在分发列表的 **open** 方法期间替换为 MQOOO_INQUIRE，因为 MQOO_OUTPUT 是此时唯一有效的 **open 选项**。但是，最好始终在使用 **open** 方法的应用程序中显式设置 MQOO_OUTPUT。
2. 如果要使用类的 **已解析队列管理器名称** 和 **已解析队列名称** 属性，请指定 MQOO_RESOLVE_NAMES。

打开状态

对象是打开 (TRUE) 还是关闭 (FALSE)。初始值为 FALSE。此属性是只读的。

先前受管对象

此类的先前对象 (无特定顺序) 与此对象具有相同的 **连接引用**。初始值为零。

队列管理器标识

队列管理器标识。此属性是只读的。

构造函数

ImqObject();

缺省构造函数。

ImqObject(const ImqObject & 对象);

复制构造函数。打开状态 将为 FALSE。

类方法 (public)

静态 MQLONG 行为 ();

返回 行为。

void setBehavior(const MQLONG 行为 = 0);

设置 行为。

对象方法 (公用)

void 运算符 = (const ImqObject & 对象);

必要时执行关闭，并从对象复制实例数据。打开状态 将为 FALSE。

ImqBoolean alterationDate(ImqString & 日期);

提供 变更日期的副本。如果成功，将返回 TRUE。

ImqString alterationDate();

返回 更改日期，而不指示任何可能的错误。

ImqBoolean alterationTime(ImqString & 时间);

提供 更改时间的副本。如果成功，将返回 TRUE。

ImqString alterationTime();

返回 更改时间，而不指示任何可能的错误。

ImqString alternateUserId() const ;

返回 备用用户标识的副本。

ImqBoolean setAlternateUserId(const char * id);

设置 备用用户标识。仅当 打开状态 为 FALSE 时，才能设置 备用用户标识。如果成功，此方法将返回 TRUE。

ImqBinary alternateSecurity 标识 () 康斯特;

返回 备用安全标识 的副本。

ImqBoolean setAlternateSecurityId(const ImqBinary & 令牌);

设置 备用安全标识。仅当 打开状态 为 FALSE 时，才能设置 备用安全标识。token 的数据长度必须为零或 MQ_SECURITY_ID_LENGTH。如果成功，将返回 TRUE。

ImqBoolean setAlternateSecurityId(const MQBYTE* token = 0);

设置 备用安全标识。token 可以为零，这与指定 MQSID_NONE 相同。如果 token 非零，那么它必须寻址二进制数据的 MQ_SECURITY_ID_LENGTH 字节。使用预定义值 (例如 MQSID_NONE) 时，可能需要进行强制类型转换以确保签名匹配; 例如，(MQBYTE *) MQSID_NONE。

仅当 打开状态 为 TRUE 时，才能设置 备用安全标识。如果成功，将返回 TRUE。

ImqBoolean setAlternateSecurityId(const unsigned char * id = 0);

设置 备用安全标识。

ImqBoolean close();

将 打开状态 设置为 FALSE。如果成功，将返回 TRUE。

MQLONG closeOptions() const ;

返回 close 选项。

void setCloseOptions(const MQLONG options);

设置 关闭选项。

ImqQueue 管理器 * connectionReference() const ;

返回 连接引用。

void setConnection 参考(ImqQueue 经理和 经理);

设置 连接引用。

void setConnectionReference(ImqQueueManager * manager = 0);

设置 连接引用。

虚拟 ImqBoolean 描述(ImqString & 描述) = 0;

提供 描述 的副本。如果成功，将返回 TRUE。

ImqString description();

返回 **description** 的副本，而不指示任何可能的错误。

虚拟 ImqBoolean 名(ImqString & 名);

提供 **name** 的副本。如果成功，将返回 TRUE。

ImqString name();

返回 **name** 的副本，而不指示任何可能的错误。

ImqBoolean setName(const char * name = 0);

设置 **名称**。仅当 **打开状态** 为 FALSE 时，才能设置 **名称**；对于 ImqQueueManager，当 **连接状态** 为 FALSE 时，才能设置 **名称**。如果成功，将返回 TRUE。

ImqObject * nextManaged 对象() const ;

返回 下一个受管对象。

ImqBoolean open();

通过在其他属性中使用 **打开选项** 和 **名称**，根据需要打开对象，将 **打开状态** 更改为 TRUE。此方法使用 **连接引用** 信息和 ImqQueueManager **connect** 方法 (如果需要) 来确保 ImqQueueManager **连接状态** 为 TRUE。它返回 **打开状态**。

ImqBoolean openFor(const MQLONG required-options = 0);

尝试确保对象是使用 **open options** 打开的，或者使用 **open options** 来保证 **required-options** 参数值隐含的行为。

如果 **required-options** 为零，那么需要输入，并且任何输入选项都足够。因此，如果 **打开选项** 已包含下列其中一项：

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

打开选项 已令人满意并且未更改；如果 **打开选项** 尚未包含任何这些选项，那么将在 **打开选项** 中设置 MQOO_INPUT_AS_Q_DEF。

如果 **required-options** 非零，那么会将必需选项添加到 **open options**；如果 **required-options** 是这些选项中的任何一个，那么将重置其他选项。

如果更改了任何 **打开选项** 并且该对象已打开，那么将临时关闭该对象并重新打开以调整 **打开选项**。

如果成功，将返回 TRUE。成功指示对象已使用相应的选项打开。

MQLONG openOptions() const ;

返回 **open options**。

ImqBoolean setOpen 选项(const MQLONG 选项);

设置 **打开选项**。仅当 **打开状态** 为 FALSE 时，才能设置 **打开选项**。如果成功，将返回 TRUE。

ImqBoolean openStatus() const ;

返回 **打开状态**。

ImqObject * previousManaged 对象() const ;

返回 先前受管对象。

ImqBoolean queueManagerIdentifier(ImqString & 标识);

提供 队列管理器标识的副本。如果成功，将返回 TRUE。

ImqString queueManagerIdentifier ();
返回 队列管理器标识，而不指示任何可能的错误。

对象方法 (protected)

虚拟 ImqBoolean closeTemporarily();
在重新打开之前安全地关闭对象。如果成功，将返回 TRUE。此方法假定 打开状态 为 TRUE。

MQHCONN connectionHandle() const ;
返回与 连接引用 关联的 MQHCONN。如果没有 连接引用，或者如果未连接管理器，那么此值为零。

ImqBoolean 查询(连接 MQLONG 内部属性, MQLONG & 值);
返回整数，其索引为 MQIA_* 值。如果发生错误，该值将设置为 MQIAV_UNDEFINED。

ImqBoolean 查询(连接 MQLONG char-属性, char * & 缓冲区, 连接大小 (t) 长度);
返回字符串，其索引为 MQCA_* 值。

注: 这两种方法都仅返回单个属性值。如果需要具有多个值的 *snapshot*，并且这些值在即时情况下彼此一致，那么 WebSphere MQ C++ 不会提供此工具，并且您必须使用带有相应参数的 MQINQ 调用。

虚拟空 openInformation 离散();
在 MQOPEN 调用之后，立即从 MQOD 数据结构的变量部分分散信息。

虚拟 ImqBoolean openInformationprepare();
在 MQOPEN 调用之前，立即为 MQOD 数据结构的变量部分准备信息，如果成功，将返回 TRUE。

ImqBoolean set(const MQLONG int-attr, const MQLONG value);
设置 WebSphere MQ 整数属性。

ImqBoolean set(const MQLONG char-attr, const char * buffer, const size_t required-length);
设置 WebSphere MQ 字符属性。

void setNextManagedObject(const ImqObject * object = 0);
设置 下一个受管对象。

注意: 仅当您确定此函数不会破坏受管对象列表时，才使用此函数。

void setPreviousManagedObject(const ImqObject * object = 0);
设置 先前受管对象。

注意: 仅当您确定此函数不会破坏受管对象列表时，才使用此函数。

对象数据 (受保护)

MQHOBJ ohobj
WebSphere MQ 对象句柄 (仅当 打开状态 为 TRUE 时有效)。

MQOD omqod
嵌入式 MQOD 数据结构。为此数据结构分配的存储量是 MQOD 版本 2 所需的存储量。检查版本号 (*omqod.Version*) 并访问其他字段，如下所示:

MQOD_VERSION_1
可以访问 *omqod* 中的所有其他字段。

MQOD_VERSION_2
可以访问 *omqod* 中的所有其他字段。

MQOD_VERSION_3
omqod.pmqod 是一个指向动态分配的较大 MQOD 的指针。无法访问 *omqod* 中的其他字段。可以访问 *omqod.pmqod* 所寻址的所有字段。

注: *omqod.pmqod.Version* 可以小于 *omqod.Version*，指示 WebSphere MQ MQI 客户机具有比 WebSphere MQ 服务器更多的功能。

原因码

- MQRC_ATTRIBUTE_LOCKED
- MQRC_INCONSISTENT_OBJECT_STATE

- MQRC_NO_CONNECTION_REFERENCE
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (来自 MQCLOSE 的原因码)
- (来自 MQCONN 的原因码)
- (来自 MQINQ 的原因码)
- (来自 MQOPEN 的原因码)
- (来自 MQSET 的原因码)

ImqProcess C++ 类

此类封装了可由触发器监视器触发的应用程序进程 (类型为 MQOT_PROCESS 的 WebSphere MQ 对象)。

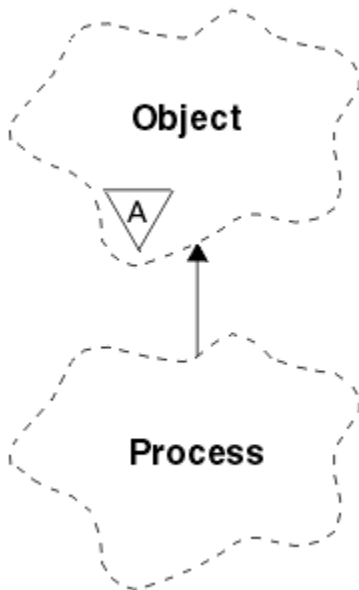


图 62: *ImqProcess* 类

- [第 1238 页的『对象属性』](#)
- [第 1238 页的『构造函数』](#)
- [第 1239 页的『对象方法 \(公用\)』](#)

对象属性

应用程序标识

应用程序进程的身份。此属性是只读的。

应用程序类型

应用程序进程的类型。此属性是只读的。

环境数据

进程的环境信息。此属性是只读的。

用户数据

进程的用户数据。此属性是只读的。

构造函数

ImqProcess();

缺省构造函数。

ImqProcess(const ImqProcess & 进程);
复制构造函数。ImqObject 打开状态 为 FALSE。

ImqProcess(const char * name);
设置 ImqObject 名称。

对象方法 (公用)

void 运算符 = (const ImqProcess & 进程);
必要时执行关闭，然后从 流程复制实例数据。ImqObject 打开状态 将为 FALSE。

ImqBoolean applicationId(ImqString & 标识);
提供 应用程序标识的副本。如果成功，将返回 TRUE。

ImqString applicationId();
返回 应用程序标识，而不指示任何可能的错误。

ImqBoolean applicationType(MQLONG & 类型);
提供 应用程序类型的副本。如果成功，将返回 TRUE。

MQLONG applicationType();
返回 应用程序类型，而不指示任何可能的错误。

ImqBoolean environmentData(ImqString & 数据);
提供 环境数据的副本。如果成功，将返回 TRUE。

ImqString environmentData();
返回 环境数据 而不指示任何可能的错误。

ImqBoolean userData(ImqString & 数据);
提供 用户数据的副本。如果成功，将返回 TRUE。

ImqString userData();
返回 用户数据，而不指示任何可能的错误。

ImqPutMessageOptions C++ 类

此类封装 MQPMO 数据结构。

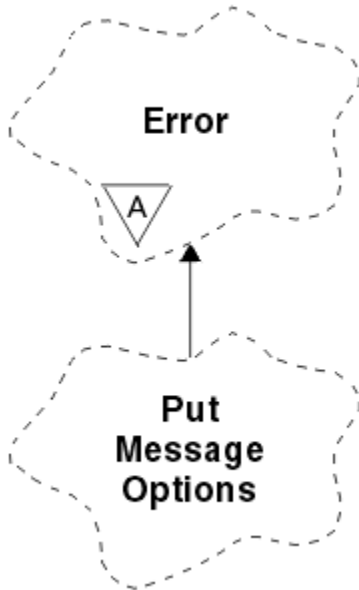


图 63: ImqPutMessageOptions 类

- [第 1240 页的『对象属性』](#)
- [第 1240 页的『构造函数』](#)
- [第 1241 页的『对象方法 \(公用\)』](#)

- [第 1241 页的『对象数据 \(受保护\)』](#)
- [第 1241 页的『原因码』](#)

对象属性

上下文参考

提供消息上下文的 `ImqQueue`。最初没有参考。

选项

put 消息选项。初始值为 `MQPMO_NONE`。可以使用以下附加值:

- `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`

记录字段

用于控制在放入消息时包含放入消息记录的标志。初始值为 `MQPMRF_NONE`。可以使用以下附加值:

- `MQPMRF_MSG_ID`
- `MQPMRF_CORREL_ID`
- `MQPMRF_GROUP_ID`
- `MQPMRF_FEEDBACK`
- `MQPMRF_ACCOUNTING_TOKEN`

`ImqMessage` 跟踪程序属性取自指定的任何字段的对象。`ImqMessage` 跟踪程序属性取自未指定的任何字段的 `ImqMessage` 对象。

已解析队列管理器名称

在放置期间确定的目标队列管理器的名称。初始值为空。此属性是只读的。

已解析队列名称

在放置期间确定的目标队列的名称。初始值为空。此属性是只读的。

同步点参与

在将消息置于同步点控制下时为 `TRUE`。

构造函数

`ImqPutMessageOptions();`

缺省构造函数。

`ImqPutMessageOptions(const ImqPutMessageOptions & 普莫);`

复制构造函数。

对象方法 (公用)

void 运算符 = (const ImqPutMessageOptions & 普莫);

从 *pmo* 复制实例数据, 替换现有实例数据。

ImqQueue * contextReference() const ;

返回 上下文引用。

void setContext 参考(const ImqQueue & 队列);

设置 上下文引用。

void setContextReference(const ImqQueue * queue = 0);

设置 上下文引用。

MQLONG 选项() const ;

返回 选项。

void setOptions(const MQLONG 选项);

设置 选项, 包括 **syncpoint** 参与 值。

MQLONG recordFields() const ;

返回 记录字段。

void setRecordFields(const MQLONG fields);

设置 记录字段。

ImqString resolvedQueueManagerName() const ;

返回 已解析的队列管理器名称的副本。

ImqString resolvedQueue 名称() const ;

返回 已解析队列名称的副本。

ImqBoolean syncPoint 参与() const ;

返回 **syncpoint** 参与 值, 如果 选项 包含 MQPMO_SYNCPOINT, 那么此值为 TRUE。

void setSyncPointParticipation(const ImqBoolean sync);

设置 **syncpoint** 参与 值。如果 *sync* 为 TRUE, 那么将更改 选项 以包含 MQPMO_SYNCPOINT 和排除 MQPMO_NO_SYNCPOINT。如果 *sync* 为 FALSE, 那么将更改 选项 以包含 MQPMO_NO_SYNCPOINT 和排除 MQPMO_SYNCPOINT。

对象数据 (受保护)

MQPMO omqpmo

MQPMO 数据结构。

原因码

- MQRC_STORAGE_NOT_AVAILABLE

ImqQueue C++ 类

此类封装消息队列 (类型为 MQOT_Q 的 WebSphere MQ 对象)。

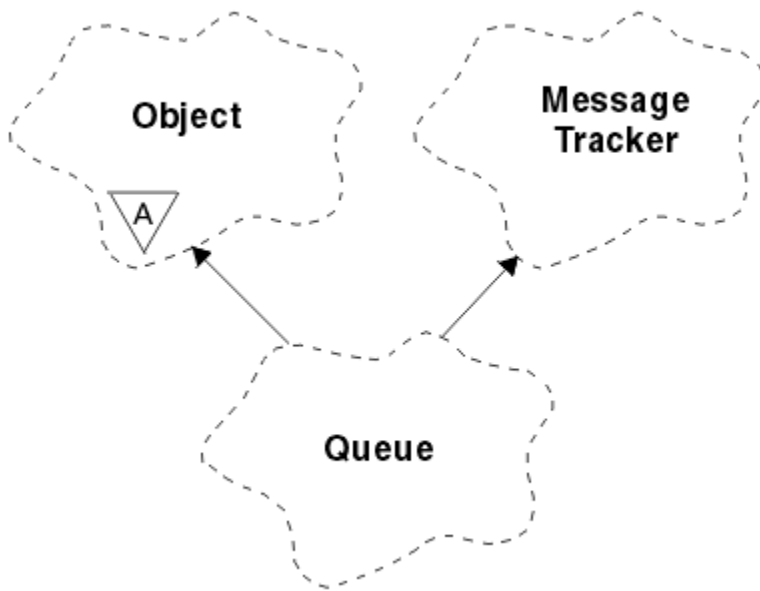


图 64: *ImqQueue* 类

此类与 [第 1185 页的表 612](#) 中列出的 MQI 调用相关。

- [第 1242 页的『对象属性』](#)
- [第 1245 页的『构造函数』](#)
- [第 1245 页的『对象方法 \(公用\)』](#)
- [第 1251 页的『对象方法 \(protected\)』](#)
- [第 1251 页的『原因码』](#)

对象属性

“回退重排队列”名称

过多的回退重排队列名称。此属性是只读的。

回退阈值

回退阈值。此属性是只读的。

基本队列名称

别名解析到的队列的名称。此属性是只读的。

集群名称

集群名称。此属性是只读的。

集群名称列表名称

集群名称列表名称。此属性是只读的。

集群工作负载等级

集群工作负载等级。此属性是只读的。

集群工作负载优先级

集群工作负载优先级。此属性是只读的。

集群工作负载使用队列

集群工作负载使用队列值。此属性是只读的。

创建日期

队列创建数据。此属性是只读的。

创建时间

队列创建时间。此属性是只读的。

当前深度

队列中的消息数。此属性是只读的。

缺省绑定

缺省绑定。此属性是只读的。

缺省输入打开选项

缺省 open-for-input 选项。此属性是只读的。

缺省持久性

缺省消息持久性。此属性是只读的。

缺省优先级

缺省消息优先级。此属性是只读的。

定义类型

队列定义类型。此属性是只读的。

深度高事件

队列深度高事件的控制属性。此属性是只读的。

深度上限

队列深度的上限。此属性是只读的。

深度下限事件

队列深度低事件的控制属性。此属性是只读的。

深度下限

队列深度的下限。此属性是只读的。

深度最大事件

队列深度最大事件数的控制属性。此属性是只读的。

分发列表参考

对可用于将消息分发到多个队列 (包括此队列) 的 ImqDistribution 列表的可引用。初始值为空。

注: 打开 ImqQueue 对象时, 将自动关闭其引用的任何打开的 ImqDistributionList 对象。

分发列表

传输队列支持分发列表的能力。此属性是只读的。

动态队列名称

动态队列名称。初始值为 AMQ.* 针对所有 Windows, UNIX 和 Linux 平台。

固化获取回退

是否硬化回退计数。此属性是只读的。

索引类型

索引类型。此属性是只读的。

禁止获取

是否允许执行 get 操作。初始值取决于队列定义。此属性仅对别名或本地队列有效。

禁止放置

是否允许执行放置操作。初始值取决于队列定义。

启动队列名称

启动队列的名称。此属性是只读的。

最大深度

队列上允许的最大消息数。此属性是只读的。

最大消息长度

此队列上任何消息的最大长度, 可以小于关联队列管理器所管理的任何队列的最大长度。此属性是只读的。

消息传递顺序

消息优先级是否相关。此属性是只读的。

下一个分布式队列

此类的下一个对象 (无特定顺序) 具有与此对象相同的 **分发列表引用**。初始值为零。

如果删除了链中的对象, 那么将更新先前的对象和下一个对象, 以使其分布式队列链接不再指向已删除的对象。

非持久消息类

放入此队列的非持久消息的可靠性级别。此属性是只读的。

打开输入计数

打开用于输入的 ImqQueue 对象数。此属性是只读的。

打开输出计数

打开用于输出的 ImqQueue 对象数。此属性是只读的。

先前的分布式队列

此类的先前对象 (无特定顺序) 具有与此对象相同的 **分发列表引用**。初始值为零。

如果删除了链中的对象, 那么将更新先前的对象和下一个对象, 以使其分布式队列链接不再指向已删除的对象。

流程名称

进程定义的名称。此属性是只读的。

队列记帐

队列的记帐信息级别。此属性是只读的。

队列管理器名称

队列所在的队列管理器 (可能是远程队列) 的名称。请勿将此处指定的队列管理器与 ImqObject **连接引用** 混淆, 后者将引用提供连接的 (本地) 队列管理器。初始值为空。

队列监视

队列的监视数据收集级别。此属性是只读的。

队列统计信息

队列的统计数据级别。此属性是只读的。

队列类型

队列类型。此属性是只读的。

远程队列管理器名称

远程队列管理器的名称。此属性是只读的。

远程队列名称

远程队列管理器上已知的远程队列的名称。此属性是只读的。

已解析队列管理器名称

已解析的队列管理器名称。此属性是只读的。

已解析队列名称

已解析的队列名称。此属性是只读的。

保留时间间隔

队列保留时间间隔。此属性是只读的。

Scope

队列定义的作用域。此属性是只读的。

服务时间间隔 (service interval)

服务时间间隔。此属性是只读的。

服务时间间隔事件 (service interval event)

服务时间间隔事件的控制属性。此属性是只读的。

可共享性

是否可以共享队列。此属性是只读的。

存储类 (storage class)

存储类。此属性是只读的。

传输队列的名称

传输队列的名称。此属性是只读的。

触发器控制

触发器控制。初始值取决于队列定义。此属性仅对本地队列有效。

触发器数据

触发器数据。初始值取决于队列定义。此属性仅对本地队列有效。

触发器深度

触发器深度。初始值取决于队列定义。此属性仅对本地队列有效。

触发器消息优先级

触发器的阈值消息优先级。初始值取决于队列定义。此属性仅对本地队列有效。

触发器类型

触发器类型。初始值取决于队列定义。此属性仅对本地队列有效。

用途

用法。此属性是只读的。

构造函数

ImqQueue();

缺省构造函数。

ImqQueue(const ImqQueue & 队列);

复制构造函数。ImqObject 打开状态 将为 FALSE。

ImqQueue(const char * name);

设置 ImqObject 名称。

对象方法 (公用)

void 运算符 = (const ImqQueue & 队列);

必要时执行关闭，然后从队列复制实例数据。ImqObject 打开状态 将为 FALSE。

ImqBoolean backoutRequeue 名称(ImqString & 名);

提供 backout 请求名称的副本。如果成功，将返回 TRUE。

ImqString backoutRequeue 名称();

返回 backout 请求名称，而不指示任何可能的错误。

ImqBoolean backoutThreshold(MQLONG & 阈值);

提供回退阈值的副本。如果成功，将返回 TRUE。

MQLONG backoutThreshold();

返回 backout threshold 值，而不指示任何可能的错误。

ImqBoolean baseQueue 名称(ImqString & 名);

提供基本队列名称的副本。如果成功，将返回 TRUE。

ImqString baseQueue 名称();

返回基本队列名称，而不指示任何可能的错误。

ImqBoolean clusterName(ImqString & 名);

提供集群名称的副本。如果成功，将返回 TRUE。

ImqString clusterName();

返回集群名称，而不指示任何可能的错误。

ImqBoolean clusterNameListName(ImqString & 名);

提供集群名称列表名称的副本。如果成功，将返回 TRUE。

ImqString clusterNameList 名称 ();

返回 cluster namelist name，而不指示任何错误。

ImqBoolean clusterWorkLoadPriority (MQLONG & priority);

提供集群工作负载优先级值的副本。如果成功，将返回 TRUE。

MQLONG clusterWorkLoadPriority ();

返回集群工作负载优先级值，而不指示任何可能的错误。

ImqBoolean clusterWorkLoadRank (MQLONG & rank);

提供集群工作负载列组值的副本。如果成功，将返回 TRUE。

MQLONG clusterWorkLoadRank ();

返回集群工作负载列组值，而不指示任何可能的错误。

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
提供集群工作负载使用队列值的副本。如果成功，将返回 TRUE。

MQLONG clusterWorkLoadUseQ ();
返回集群工作负载使用队列值，而不指示任何可能的错误。

ImqBoolean creationDate(ImqString & 日期);
提供 创建日期 的副本。如果成功，将返回 TRUE。

ImqString creationDate();
返回 创建日期，而不指示任何可能的错误。

ImqBoolean creationTime(ImqString & 时间);
提供 创建时间 的副本。如果成功，将返回 TRUE。

ImqString creationTime();
返回 创建时间，而不指示任何可能的错误。

ImqBoolean currentDepth(MQLONG & 深度);
提供 当前深度 的副本。如果成功，将返回 TRUE。

MQLONG currentDepth();
返回 当前深度，而不指示任何可能的错误。

ImqBoolean defaultInputOpenOption(MQLONG & 选项);
提供 缺省输入打开选项 的副本。如果成功，将返回 TRUE。

MQLONG defaultInputOpenOption();
返回 缺省输入打开选项，而不指示任何可能的错误。

ImqBoolean defaultPersistence(MQLONG & 持久性);
提供 缺省持久性 的副本。如果成功，将返回 TRUE。

MQLONG defaultPersistence();
返回 缺省持久性，而不指示任何可能的错误。

ImqBoolean defaultPriority(MQLONG & 优先级);
提供 缺省优先级 的副本。如果成功，将返回 TRUE。

MQLONG defaultPriority();
返回 缺省优先级，而不指示任何可能的错误。

ImqBoolean defaultBind(MQLONG & 绑定);
提供 缺省绑定 的副本。如果成功，将返回 TRUE。

MQLONG defaultBind();
返回 缺省绑定，而不指示任何可能的错误。

ImqBoolean definitionType(MQLONG & 类型);
提供 定义类型 的副本。如果成功，将返回 TRUE。

MQLONG definitionType();
返回 定义类型，而不指示任何可能的错误。

ImqBoolean depthHigh 事件(MQLONG & 事件);
提供 深度高事件的启用状态的副本。如果成功，将返回 TRUE。

MQLONG depthHigh 事件();
返回 深度高事件 的启用状态，而不指示任何可能的错误。

ImqBoolean depthHigh 限制(MQLONG & 限制);
提供 深度上限 的副本。如果成功，将返回 TRUE。

MQLONG depthHigh 限制();
返回 深度上限 值，而不指示任何可能的错误。

ImqBoolean depthLow 事件(MQLONG & 事件);
提供 深度低事件的启用状态的副本。如果成功，将返回 TRUE。

MQLONG depthLow 事件();
返回 深度低事件 的启用状态，而不指示任何可能的错误。

ImqBoolean depthLow 限制(MQLONG & 限制);
提供 深度下限 的副本。如果成功，将返回 TRUE。

MQLONG depthLow 限制();

返回 深度下限 值，而不指示任何可能的错误。

ImqBoolean depthMaximum 事件(MQLONG & 事件);

提供 深度最大事件的启用状态的副本。如果成功，将返回 TRUE。

MQLONG depthMaximumEvent();

返回 深度最大事件 的启用状态，而不指示任何可能的错误。

ImqDistributionList * distributionListReference() const ;

返回 分发列表引用。

void setDistributionListReference(ImqDistributionList & 列表);

设置 分发列表引用。

void setDistributionListReference(ImqDistributionList * list = 0);

设置 分发列表引用。

ImqBoolean distributionLists(MQLONG & 支持);

提供 分发列表 值的副本。如果成功，将返回 TRUE。

MQLONG distributionLists();

返回 分布列表 值，而不指示任何可能的错误。

ImqBoolean setDistributionList(const MQLONG 支持);

设置 分布列表 值。如果成功，将返回 TRUE。

ImqString dynamicQueue 名称() const ;

返回 动态队列名称的副本。

ImqBoolean setDynamicQueueName(const char * name);

设置 动态队列名称。仅当 ImqObject 打开状态为 FALSE 时，才能设置 动态队列名称。如果成功，将返回 TRUE。

ImqBoolean 获取(ImqMessage & 消息, ImqGetMessageOptions & 选项);

使用指定的 选项从队列中检索消息。如果需要，调用 ImqObject **openFor** 方法以确保 ImqObject **open options** 包含 MQOO_INPUT_ * 值之一或 MQOO_BROWSE 值，具体取决于 *options*。如果 *msg* 对象具有 ImqCache 自动缓冲区，那么缓冲区将增大以容纳检索到的任何消息。在检索之前，将针对 *msg* 对象调用 **clearMessage** 方法。

如果成功，此方法将返回 TRUE。

注: 如果 ImqObject 原因码为 MQRC_TRUNCATED_MSG_FAILED，那么方法调用的结果为 FALSE，即使此原因码分类为警告也是如此。如果接受截断的消息，那么 ImqCache 消息长度 将反映截断的长度。在任一情况下，ImqMessage 消息总长度 指示可用的字节数。

ImqBoolean 获取(ImqMessage & 消息);

对于先前的方法，除了使用缺省 get 消息选项之外。

ImqBoolean 获取(ImqMessage & 消息, ImqGetMessageOptions & 选项, 连接大小 (t) 缓冲区大小);

对于前两种方法，除了指示覆盖 *buffer-size* 之外。如果 *msg* 对象采用 ImqCache 自动缓冲区，那么在检索消息之前将在 *msg* 对象上调用 **resizeBuffer** 方法，并且缓冲区不会进一步增大以容纳任何更大的消息。

ImqBoolean 获取(ImqMessage & 消息, 连接大小 (t) 缓冲区大小);

对于先前的方法，除了使用缺省 get 消息选项之外。

ImqBoolean hardenGet 回退(MQLONG & 哈登);

提供 **harden get backout** 值的副本。如果成功，将返回 TRUE。

MQLONG hardenGet 回退();

返回 **harden get backout** 值，而不指示任何可能的错误。

ImqBoolean indexType(MQLONG & 类型);

提供 索引类型的副本。如果成功，将返回 TRUE。

MQLONG indexType();

返回 索引类型，而不指示任何可能的错误。

ImqBoolean inhibitGet(MQLONG & 抑制);

提供 禁止获取 值的副本。如果成功，将返回 TRUE。

MQLONG inhibitGet();

返回 **禁止获取** 值，而不指示任何可能的错误。

ImqBoolean setInhibitGet(const MQLONG 禁止);

设置 **禁止获取** 值。如果成功，将返回 TRUE。

ImqBoolean inhibitPut(MQLONG & 抑制);

提供 **禁止放置** 值的副本。如果成功，将返回 TRUE。

MQLONG inhibitPut();

返回 **禁止放置** 值，而不指示任何可能的错误。

ImqBoolean setInhibitPut(const MQLONG 禁止);

设置 **禁止放置** 值。如果成功，将返回 TRUE。

ImqBoolean initiationQueue 名称(ImqString & 名);

提供 **启动队列名称** 的副本。如果成功，将返回 TRUE。

ImqString initiationQueue 名称();

返回 **启动队列名称**，而不指示任何可能的错误。

ImqBoolean maximumDepth(MQLONG & 深度);

提供 **最大深度** 的副本。如果成功，将返回 TRUE。

MQLONG maximumDepth();

返回 **最大深度**，而不指示任何可能的错误。

ImqBoolean maximumMessage 长度(MQLONG & 长度);

提供 **最大消息长度** 的副本。如果成功，将返回 TRUE。

MQLONG maximumMessage 长度();

返回 **最大消息长度**，而不指示任何可能的错误。

ImqBoolean messageDelivery 序列(MQLONG & 序列);

提供 **消息传递序列** 的副本。如果成功，将返回 TRUE。

MQLONG messageDelivery 序列();

返回 **message delivery sequence** 值，而不指示任何可能的错误。

ImqQueue * nextDistributed 队列() const ;

返回 **下一个分布式队列**。

ImqBoolean nonPersistentMessageClass (MQLONG & monq);

提供 **非持久消息类值** 的副本。如果成功，将返回 TRUE。

MQLONG nonPersistentMessageClass ();

返回 **非持久消息类值**，而不指示任何可能的错误。

ImqBoolean openInput 计数(MQLONG & 计数);

提供 **打开输入计数** 的副本。如果成功，将返回 TRUE。

MQLONG openInput 计数();

返回 **打开输入计数**，而不指示任何可能的错误。

ImqBoolean openOutput 计数(MQLONG & 计数);

提供 **打开输出计数** 的副本。如果成功，将返回 TRUE。

MQLONG openOutput 计数();

返回 **打开输出计数**，而不指示任何可能的错误。

ImqQueue * previousDistributed 队列() const ;

返回 **先前的分布式队列**。

ImqBoolean processName(ImqString & 名);

提供 **流程名称** 的副本。如果成功，将返回 TRUE。

ImqString processName();

返回 **进程名称**，而不指示任何可能的错误。

ImqBoolean 放(ImqMessage & 消息);

使用缺省 put 消息选项将消息放入队列中。必要时使用 ImqObject **openFor** 方法以确保 ImqObject **打开选项** 包含 MQOO_OUTPUT。

如果成功，此方法将返回 TRUE。

ImqBoolean 放(ImqMessage & 消息, ImqPutMessageOptions & 普莫);

使用指定的 *pmo* 将消息放入队列。根据需要使用 ImqObject **openFor** 方法，以确保 ImqObject 打开选项包含 MQOO_OUTPUT，并且 (如果 *pmo* 选项 包含 MQPMO_PASS_IDENTITY_CONTEXT，MQPMO_PASS_ALL_CONTEXT，MQPMO_SET_IDENTITY_CONTEXT 或 MQPMO_SET_IDENTITY_CONTEXT) 相应的 MQOO_*_CONTEXT 值。

如果成功，此方法将返回 TRUE。

注: 如果 *pmo* 包含 上下文引用，那么将在必要时打开引用的对象以提供上下文。

ImqBoolean queueAccounting (MQLONG & acctq);

提供队列记帐值的副本。如果成功，将返回 TRUE。

MQLONG queueAccounting ();

返回队列记帐值，而不指示任何可能的错误。

ImqString queueManagerName() const ;

返回 队列管理器名称。

ImqBoolean setQueueManagerName(const char * name);

设置 队列管理器名称。仅当 ImqObject 打开状态为 FALSE 时，才能设置 队列管理器名称。如果成功，此方法将返回 TRUE。

ImqBoolean queueMonitoring (MQLONG & monq);

提供队列监视值的副本。如果成功，将返回 TRUE。

MQLONG queueMonitoring ();

返回队列监视值，而不指示任何可能的错误。

ImqBoolean queueStatistics (MQLONG 和 statq);

提供队列统计信息值的副本。如果成功，将返回 TRUE。

MQLONG queueStatistics ();

返回队列统计信息值，而不指示任何可能的错误。

ImqBoolean queueType(MQLONG & 类型);

提供 队列类型 值的副本。如果成功，将返回 TRUE。

MQLONG queueType();

返回 队列类型，而不指示任何可能的错误。

ImqBoolean remoteQueueManagerName(ImqString & 名);

提供 远程队列管理器名称的副本。如果成功，将返回 TRUE。

ImqString remoteQueueManagerName();

返回 远程队列管理器名称，而不指示任何可能的错误。

ImqBoolean remoteQueue 名称(ImqString & 名);

提供 远程队列名称的副本。如果成功，将返回 TRUE。

ImqString remoteQueue 名称();

返回 远程队列名称，而不指示任何可能的错误。

ImqBoolean resolvedQueueManagerName(ImqString & 名);

提供 已解析的队列管理器名称的副本。如果成功，将返回 TRUE。

注: 除非 MQOO_RESOLVE_NAMES 位于 ImqObject 打开选项中，否则此方法将失败。

ImqString resolvedQueueManagerName();

返回 已解析的队列管理器名称，而不指示任何可能的错误。

ImqBoolean resolvedQueueName(ImqString & 名);

提供 已解析队列名称的副本。如果成功，将返回 TRUE。

注: 除非 MQOO_RESOLVE_NAMES 位于 ImqObject 打开选项中，否则此方法将失败。

ImqString resolvedQueueName ();

返回 已解析的队列名称，而不指示任何可能的错误。

ImqBoolean retentionInterval(MQLONG & 间隔);
提供 保留时间间隔 的副本。如果成功，将返回 TRUE。

MQLONG retentionInterval();
返回 保留时间间隔，而不指示任何可能的错误。

ImqBoolean 范围(MQLONG & 范围);
提供 作用域的副本。如果成功，将返回 TRUE。

MQLONG 作用域();
返回 作用域，而不指示任何可能的错误。

ImqBoolean serviceInterval(MQLONG & 间隔);
提供 服务时间间隔 的副本。如果成功，将返回 TRUE。

MQLONG serviceInterval();
返回 服务时间间隔，而不指示任何可能的错误。

ImqBoolean serviceInterval 事件(MQLONG & 事件);
提供 服务时间间隔事件的启用状态的副本。如果成功，将返回 TRUE。

MQLONG serviceInterval 事件();
返回 服务时间间隔事件的启用状态，而不指示任何可能的错误。

ImqBoolean 可共享性(MQLONG & 可共享性);
提供 shareability 值的副本。如果成功，将返回 TRUE。

MQLONG 可共享性();
返回 shareability 值，而不指示任何可能的错误。

ImqBoolean storageClass(ImqString & 类);
提供 存储类的副本。如果成功，将返回 TRUE。

ImqString storageClass();
返回 存储类，而不指示任何可能的错误。

ImqBoolean transmissionQueue 名称(ImqString & 名);
提供 传输队列名称的副本。如果成功，将返回 TRUE。

ImqString transmissionQueue 名称();
返回 传输队列名称，而不指示任何可能的错误。

ImqBoolean triggerControl(MQLONG & 控制);
提供 trigger control 值的副本。如果成功，将返回 TRUE。

MQLONG triggerControl();
返回 trigger control 值，而不指示任何可能的错误。

ImqBoolean setTriggerControl(const MQLONG control);
设置 trigger control 值。如果成功，将返回 TRUE。

ImqBoolean triggerData(ImqString & 数据);
提供 触发器数据的副本。如果成功，将返回 TRUE。

ImqString triggerData();
返回 触发器数据 的副本，而不指示任何可能的错误。

ImqBoolean setTriggerData(const char * data);
设置 触发器数据。如果成功，将返回 TRUE。

ImqBoolean triggerDepth(MQLONG & 深度);
提供 触发器深度的副本。如果成功，将返回 TRUE。

MQLONG triggerDepth();
返回 触发器深度，而不指示任何可能的错误。

ImqBoolean setTriggerDepth(const MQLONG depth);
设置 触发器深度。如果成功，将返回 TRUE。

ImqBoolean triggerMessage 优先级(MQLONG & 优先级);
提供 触发器消息优先级的副本。如果成功，将返回 TRUE。

MQLONG triggerMessage 优先级();
返回 触发器消息优先级，而不指示任何可能的错误。

ImqBoolean setTriggerMessagePriority(const MQLONG *priority*);

设置 触发器消息优先级。如果成功，将返回 TRUE。

ImqBoolean triggerType(MQLONG & 类型);

提供 触发器类型的副本。如果成功，将返回 TRUE。

MQLONG triggerType();

返回 触发器类型，而不指示任何可能的错误。

ImqBoolean setTriggerType(const MQLONG *type*);

设置 触发器类型。如果成功，将返回 TRUE。

ImqBoolean 用法(MQLONG & 用法);

提供 **usage** 值的副本。如果成功，将返回 TRUE。

MQLONG 用法();

返回 **usage** 值，而不指示任何可能的错误。

对象方法 (protected)

void setNextDistributedQueue(ImqQueue * *queue* = 0);

设置 下一个分布式队列。

注意: 仅当您确定此函数不会破坏分布式队列列表时，才使用此函数。

void setPreviousDistributedQueue(ImqQueue * *queue* = 0);

设置 先前的分布式队列。

注意: 仅当您确定此函数不会破坏分布式队列列表时，才使用此函数。

原因码

- 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
- (来自 MQGET 的原因码)
- (来自 MQPUT 的原因码)

ImqQueueManager C++ 类

此类封装队列管理器 (类型为 MQOT_Q_MGR 的 WebSphere MQ 对象)。

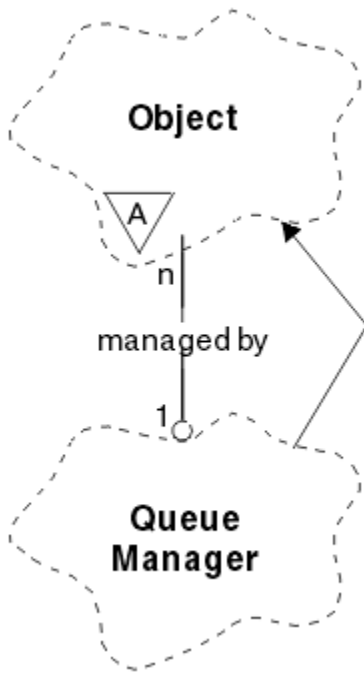


图 65: *ImqQueueManager* 类

此类与第 1187 页的『*ImqQueueManager* 交叉引用』中列出的 MQI 调用相关。并非所有列出的方法都适用于所有平台; 请参阅 [ALTER QMGR](#) 以获取更多详细信息。

- [第 1252 页的『类属性』](#)
- [第 1253 页的『对象属性』](#)
- [第 1257 页的『构造函数』](#)
- [第 1257 页的『析构函数』](#)
- [第 1257 页的『类方法 \(public\)』](#)
- [第 1257 页的『对象方法 \(公用\)』](#)
- [第 1266 页的『对象方法 \(protected\)』](#)
- [第 1266 页的『对象数据 \(受保护\)』](#)
- [第 1266 页的『原因码』](#)

类属性

行为 (behavior)

控制隐式连接和断开连接的行为。

IMQ_EXPL_DISC_BACKOUT (0L)

对 **disconnect** 方法的显式调用意味着回退。此属性与 **IMQ_EXPL_DISC_COMMIT** 互斥。

IMQ_EXPL_DISC_COMMIT (1L)

对 **disconnect** 方法的显式调用意味着落实 (缺省值)。此属性与 **IMQ_EXPL_DISC_BACKOUT** 互斥。

IMQ_IMPL_CONN (2L)

允许隐式连接 (缺省值)。

IMQ_IMPL_DISC_BACKOUT (0L)

隐式调用 **disconnect** 方法 (可能在对象破坏期间发生) 意味着回退。此属性与 **IMQ_IMPL_DISC_COMMIT** 互斥。

IMQ_IMPL_DISC_COMMIT (4L)

对 **disconnect** 方法的隐式调用 (可能在对象破坏期间发生) 意味着落实 (缺省值)。此属性与 **IMQ_IMPL_DISC_BACKOUT** 互斥。

在 WebSphere MQ V7.0 及更高版本中，使用隐式连接的 C++ 应用程序需要指定 `IMQ_IMPL_CONN` 以及类 `ImqQueueManager` 的对象上的 `setBehavior()` 方法中提供的任何其他选项。如果应用程序未使用 `setBehavior()` 方法来显式设置行为选项，例如，

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

此更改不会影响您，因为缺省情况下已启用 `MQ_IMPL_CONN`。

如果应用程序显式设置行为选项，例如，

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

您需要在 `setBehavior()` 方法中包含 `IMQ_IMPL_CONN`，如下所示，以允许应用程序完成隐式连接：

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

对象属性

记帐连接覆盖

允许应用程序覆盖 MQI 记帐和队列记帐 `values.This` 属性是只读属性。

记帐周期

写入中间记帐记录之前的时间长度 (以秒计)。此属性是只读的。

活动记录

控制活动报告的生成。此属性是只读的。

采用新的 MCA 检查

检查元素以确定在检测到与已处于活动状态的 MCA 同名的新入站通道时是否应采用 MCA。此属性是只读的。

采用新的 MCA 类型

当检测到与采用新 `mca` 检查参数匹配的新入站通道请求时，是否应该自动重新启动特定通道类型的 MCA 的孤立实例。此属性是只读的。

认证类型

指示正在执行的认证的类型。

权限事件

控制权限事件。此属性是只读的。

开始选项

适用于 `begin` 方法的选项。初始值为 `MQBO_NONE`。

网桥事件

是否生成 IMS 网桥事件。此属性是只读的。

通道自动定义

通道自动定义值。此属性是只读的。

通道自动定义事件

通道自动定义事件值。此属性是只读的。

通道自动定义出口

通道自动定义出口名称。此属性是只读的。

通道事件 (channel event)

是否生成通道事件。此属性是只读的。

通道启动程序适配器

用于处理 WebSphere MQ 调用的适配器子任务数。此属性是只读的。

通道启动程序控制方式

在启动队列管理器时是否应自动启动通道启动程序。此属性是只读的。

通道启动程序分派器

要用于通道启动程序的分派器数。此属性是只读的。

通道启动程序跟踪自动启动

通道启动程序跟踪是否应该自动启动。此属性是只读的。

通道启动程序跟踪表大小

通道启动程序的跟踪数据空间大小 (MB)。此属性是只读的。

通道监视

控制通道联机监视数据的收集。此属性是只读的。

通道引用

对要在客户机连接期间使用的通道定义的引用。连接时，此属性可以设置为 null，但不能更改为任何其他值。初始值为空。

通道统计

控制通道的统计数据收集。此属性是只读的。

字符集

编码字符集标识 (CCSID)。此属性是只读的。

集群发送方监视

控制自动定义的集群发送方通道的联机监视数据收集。此属性是只读的。

集群发送方统计

控制自动定义的集群发送方通道的统计数据收集。此属性是只读的。

集群工作负载数据

集群工作负载出口数据。此属性是只读的。

集群工作负载出口

集群工作负载出口名称。此属性是只读的。

集群工作负载长度

集群工作负载长度。此属性是只读的。

集群工作负载 mru

集群工作负载最近使用的通道值。此属性是只读的。

集群工作负载使用队列

集群工作负载使用队列值。此属性是只读的。

命令事件 (command event)

是否生成命令事件。此属性是只读的。

命令输入队列名称

系统命令输入队列名称。此属性是只读的。

命令级别

队列管理器支持的命令级别。此属性是只读的。

命令服务器控制

在启动队列管理器时是否应自动启动命令服务器。此属性是只读的。

连接选项

适用于 **connect** 方法的选项。初始值为 MQCNO_NONE。根据平台的不同，可以使用以下附加值：

- 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

连接标识

允许 MQ 可靠地标识应用程序的唯一标识。

连接状态

连接到队列管理器时为 TRUE。此属性是只读的。

连接标记

要与连接关联的标记。仅当未连接时才能设置此属性。初始值为空。

加密硬件

加密硬件的配置详细信息。对于 MQ MQI 客户机连接。

死信队列名称

死信队列的名称。此属性是只读的。

缺省传输队列名称

缺省传输队列名称。此属性是只读的。

分发列表

队列管理器支持分发列表的能力。

DNS 组

使用 "工作负载管理器动态域名服务" 支持时, 处理队列共享组的进站传输的 TCP 侦听器应加入的组的名称。此属性是只读的。

丹

处理队列共享组的进站传输的 TCP 侦听器是否应该向 Dynamic Domain Name Services 的工作负载管理器注册。此属性是只读的。

第一个认证记录

类 ImqAuthentication 记录的一个或多个对象中的第一个对象 (无特定顺序), 其中 ImqAuthentication 记录连接引用对此对象进行寻址。对于 MQ MQI 客户机连接。

第一受管对象

类 ImqObject 的第一个或多个对象 (不按特定顺序), 其中 ImqObject 连接引用 对此对象进行寻址。初始值为零。

禁止事件

控制禁止事件。此属性是只读的。

IP 地址版本

要用于通道连接的 IP 协议 (IPv4 或 IPv6)。此属性是只读的。

密钥库 (key repository)

存储密钥和证书的密钥数据库文件的位置。对于 WebSphere MQ MQI 客户机连接。

密钥重置计数

在重新协商密钥之前, 在 SSL 对话中发送和接收的未加密字节数。此属性仅适用于使用 MQCONN 的客户机连接。另请参阅 [SSL 密钥重置计数 \(ssl key reset count\)](#)。

侦听器计时器

当发生 APPC 或 TCP/IP 故障时, WebSphere MQ 尝试重新启动侦听器之间的时间间隔 (以秒计)。此属性是只读的。

本地事件

控制本地事件。此属性是只读的。

记录器事件

控制是否生成恢复日志事件。此属性是只读的。

LU 组名

处理队列共享组的进站传输的 LU 6.2 侦听器应该使用的通用 LU 名。此属性是只读的。

LU 名

要用于出站 LU 6.2 传输的 LU 的名称。此属性是只读的。

lu62 臂后缀

SYS1.PARMLIB 成员 APPCPMxx, 用于指定此通道启动程序的 LUADD。此属性是只读的。

lu62 通道

可以是当前通道或可以连接的客户机 (使用 LU 6.2 传输协议) 的最大通道数。此属性是只读的。

最大活动通道数

在任一时刻可以处于活动状态的最大通道数。此属性是只读的。

最大通道数

当前最大通道数 (包括带有已连接的客户机的服务器连接通道)。此属性是只读的。

最大句柄数

最大句柄数。此属性是只读的。

最大消息长度

此队列管理器管理的任何队列上的任何消息的最大可能长度。此属性是只读的。

最高优先级

最大消息优先级。此属性是只读的。

未落实消息的最大数目

单元或工作中未落实的最大消息数。此属性是只读的。

MQI 记帐

控制 MQI 数据的记帐信息的收集。此属性是只读的。

MQI 统计

控制队列管理器统计监视信息的收集。此属性是只读的。

最大出站端口数

绑定传出通道时要使用的端口号范围的较高端。此属性是只读的。

出站端口最小值

绑定传出通道时要使用的端口号范围的下端。此属性是只读的。

密码

与用户标识关联的密码

性能事件 (performance event)

控制性能事件。此属性是只读的。

platform

队列管理器所在的平台。此属性是只读的。

队列记帐

控制队列的记帐信息的收集。此属性是只读的。

队列监视

控制队列联机监视数据的收集。此属性是只读的。

队列统计信息

控制队列的统计数据收集。此属性是只读的。

接收超时

TCP/IP 消息通道在返回到不活动状态之前等待从其伙伴接收数据 (包括脉动信号) 的大约时间长度。此属性是只读的。

接收超时最小值

TCP/IP 通道在返回到不活动状态之前等待从其伙伴接收数据 (包括脉动信号) 的最短时间。此属性是只读的。

接收超时类型

应用于 **receive timeout** 的限定符。此属性是只读的。

远程事件

控制远程事件。此属性是只读的。

存储库名称

存储库名称。此属性是只读的。

存储库名称列表

存储库名称列表名称。此属性是只读的。

共享队列管理器名称

ObjectQMgr 名称是队列共享组中的另一个队列管理器的共享队列的 MQOPEN 是否应解析为本地队列管理器上的共享队列的打开。此属性是只读的。

ssl 事件

是否生成 SSL 事件。此属性是只读的。

需要 SSL FIPS

如果在 WebSphere MQ 软件中执行密码术, 那么是否仅应使用 FIPS 认证的算法。此属性是只读的。

SSL 密钥重新设置计数

在重新协商密钥之前，在 SSL 对话中发送和接收的未加密字节数。此属性是只读的。

启动-停止事件

控制启动-停止事件。此属性是只读的。

统计时间间隔

统计信息监视数据写入监视队列的频率。此属性是只读的。

同步点的可用性

提供同步点参与。此属性是只读的。

注: 在 IBM i 平台上不支持队列管理器协调的全局工作单元。

tcp 通道

可以是当前通道或可以连接的客户机 (使用 TCP/IP 传输协议) 的最大通道数。此属性是只读的。

TCP 保持活动

是否使用 TCP KEEPALIVE 工具来检查连接的另一端是否仍然可用。此属性是只读的。

TCP 名称

要使用的唯一或缺省 TCP/IP 系统的名称，具体取决于 **tcp 堆栈类型** 的值。此属性是只读的。

TCP 堆栈类型

是允许通道启动程序仅使用 **tcp name** 中指定的 TCP/IP 地址空间，还是可以绑定到任何选定的 TCP/IP 地址。此属性是只读的。

跟踪路由记录

控制路由跟踪信息的记录。此属性是只读的。

触发器时间间隔

触发时间间隔。此属性是只读的。

用户标识

在 UNIX and Linux 平台上，应用程序的实际用户标识。在 Windows 平台上，应用程序的用户标识。

构造函数

ImqQueueManager ();

缺省构造函数。

ImqQueue 经理 (const ImqQueueManager & 经理);

复制构造函数。连接状态 将为 FALSE。

ImqQueueManager (const char * name);

将 ImqObject name 设置为 name。

析构函数

当 ImqQueueManager 对象被破坏时，它将自动断开连接。

类方法 (public)

静态 MQLONG 行为 ();

返回 行为。

void setBehavior(const MQLONG 行为 = 0);

设置 行为。

对象方法 (公用)

void operator = (const ImqQueueManager & 经理);

必要时断开连接，并从 mgr 复制实例数据。连接状态 为 FALSE。

ImqBoolean accountingConn 覆盖 (MQLONG 和 statint);

提供记帐连接覆盖值的副本。如果成功，将返回 TRUE。

MQLONG accountingConn 覆盖 ();

返回记帐连接覆盖值，而不指示任何可能的错误。

ImqBoolean accountingInterval (MQLONG 和 statint);

提供记帐时间间隔值的副本。如果成功，将返回 TRUE。

MQLONG accountingInterval ();

返回记帐时间间隔值，而不指示任何可能的错误。

ImqBoolean activityRecording (MQLONG 和 rec);

提供活动记录值的副本。如果成功，将返回 TRUE。

MQLONG activityRecording ();

返回活动记录值，而不指示可能的错误。

ImqBoolean adoptNewMCACheck (MQLONG & check);

提供采用新 MCA 检查值的副本。如果成功，将返回 TRUE。

MQLONG adoptNewMCACheck ();

返回采用新的 MCA 检查值，而不指示任何可能的错误。

ImqBoolean adoptNewMCAType (MQLONG & type);

提供采用新 MCA 类型的副本。如果成功，将返回 TRUE。

MQLONG adoptNewMCAType ();

返回采用新的 MCA 类型，而不指示任何可能的错误。

QLONG authenticationType () 康斯特;

返回认证类型。

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);

设置认证类型。

ImqBoolean authorityEvent (MQLONG & 事件);

提供 权限事件的启用状态的副本。如果成功，将返回 TRUE。

MQLONG authorityEvent();

返回 权限事件 的启用状态，而不指示任何可能的错误。

ImqBoolean backout ();

回退未落实的更改。如果成功，将返回 TRUE。

ImqBoolean begin ();

开始工作单元。begin options 会影响此方法的行为。如果成功，它将返回 TRUE，但即使底层 MQBEGIN 调用返回 MQRC_NO_EXTERNAL_MEMBERS 或 MQRC_PARTICIPANT_NOT_AVAILABLE (两者都与 MQCC_WARNING 相关联)，它也将返回 TRUE。

MQLONG beginOptions() 康斯特;

返回 begin options。

void setBeginOptions (const MQLONG options = MQBO_NONE);

设置 开始选项。

ImqBoolean bridgeEvent (MQLONG & event);

提供网桥事件值的副本。如果成功，将返回 TRUE。

MQLONG bridgeEvent ();

返回网桥事件值，而不指示任何可能的错误。

ImqBoolean channelAutoDefinition (MQLONG & 值);

提供 channel auto definition 值的副本。如果成功，将返回 TRUE。

MQLONG channelAuto 定义 ();

返回 channel auto definition 值，而不指示任何可能的错误。

ImqBoolean channelAutoDefinitionEvent (MQLONG & 值);

提供 通道自动定义事件 值的副本。如果成功，将返回 TRUE。

MQLONG channelAutoDefinitionEvent();

返回 通道自动定义事件 值，而不指示任何可能的错误。

ImqBoolean channelAutoDefinitionExit (ImqString & 名);

提供 通道自动定义出口 名称的副本。如果成功，将返回 TRUE。

ImqString channelAutoDefinitionExit();

返回 通道自动定义出口 名称，而不指示任何可能的错误。

ImqBoolean channelEvent (MQLONG & event);

提供通道事件值的副本。如果成功，将返回 TRUE。

MQLONG channelEvent();

返回通道事件值，而不指示任何可能的错误。

MQLONG channelInitiator 适配器 ();

返回通道启动程序适配器值，而不指示任何可能的错误。

ImqBoolean channelInitiator 适配器 (MQLONG 和适配器);

提供通道启动程序适配器值的副本。如果成功，将返回 TRUE。

MQLONG channelInitiator 控制 ();

返回通道启动程序启动值，而不指示任何可能的错误。

ImqBoolean channelInitiatorControl (MQLONG & init);

提供通道启动程序控制启动值的副本。如果成功，将返回 TRUE。

MQLONG channelInitiator 分派器 ();

返回通道启动程序分派器值，而不指示任何可能的错误。

ImqBoolean channelInitiator 分派器 (MQLONG 和分派器);

提供通道启动程序分派器值的副本。如果成功，将返回 TRUE。

MQLONG channelInitiatorTraceAuto 启动 ();

返回通道启动程序跟踪自动启动值，而不指示任何可能的错误。

ImqBoolean channelInitiatorTraceAutoStart (MQLONG & auto);

提供通道启动程序跟踪自动启动值的副本。如果成功，将返回 TRUE。

MQLONG channelInitiatorTraceTable 大小 ();

返回通道启动程序跟踪表大小值，而不指示任何可能的错误。

ImqBoolean channelInitiatorTraceTable 大小 (MQLONG 和大小);

提供通道启动程序跟踪表大小值的副本。如果成功，将返回 TRUE。

ImqBoolean channelMonitoring (MQLONG 和 monchl);

提供通道监视值的副本。如果成功，将返回 TRUE。

MQLONG channelMonitoring ();

返回通道监视值，而不指示任何可能的错误。

ImqBoolean channelReference(ImqChannel * & Pchannel);

提供 通道引用的副本。如果 通道引用 无效，请将 *pchannel* 设置为空。如果成功，此方法将返回 TRUE。

ImqChannel * channelReference();

返回 通道引用 ，而不指示任何可能的错误。

ImqBoolean setChannelReference(ImqChannel & 通道);

设置 通道引用。如果成功，此方法将返回 TRUE。

ImqBoolean setChannel 参考 (ImqChannel * channel = 0);

设置或重置 通道引用。如果成功，此方法将返回 TRUE。

ImqBoolean channelStatistics (MQLONG 和 statchl);

提供通道统计信息值的副本。如果成功，将返回 TRUE。

MQLONG channelStatistics ();

返回通道统计信息值，而不指示任何可能的错误。

ImqBoolean characterSet (MQLONG & 奇西德);

提供 字符集的副本。如果成功，将返回 TRUE。

MQLONG characterSet();

返回 字符集的副本，而不指示任何可能的错误。

MQLONG clientSslKeyReset 计数 () 康斯特;

返回客户机连接上使用的 SSL 密钥重置计数值。

void setClientSslKeyResetCount(const MQLONG 计数);
设置在客户机连接上使用的 SSL 密钥重置计数。

ImqBoolean clusterSender 监视 (MQLONG 和 monacls);
提供集群发送方监视缺省值的副本。如果成功，将返回 TRUE。

MQLONG clusterSender 监视 ();
返回集群发送方监视缺省值，而不指示任何可能的错误。

ImqBoolean clusterSender 统计信息 (MQLONG 和 statacls);
提供集群发送方统计信息值的副本。如果成功，将返回 TRUE。

MQLONG clusterSender 统计信息 ();
返回集群发送方统计信息值，而不指示任何可能的错误。

ImqBoolean clusterWorkloadData(ImqString & 数据);
提供 集群工作负载出口数据 的副本。如果成功，将返回 TRUE。

ImqString clusterWorkload 数据 ();
返回 集群工作负载出口数据 ，而不指示任何可能的错误。

ImqBoolean clusterWorkloadExit(ImqString & 名);
提供 集群工作负载出口名称 的副本。如果成功，将返回 TRUE。

ImqString clusterWorkloadExit ();
返回 集群工作负载出口名称 ，而不指示任何可能的错误。

ImqBoolean clusterWorkload 长度 (MQLONG 和 长度);
提供 集群工作负载长度 的副本。如果成功，将返回 TRUE。

MQLONG clusterWorkload 长度 ();
返回 集群工作负载长度 ，而不指示任何可能的错误。

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);
提供最近使用的集群工作负载通道值的副本。如果成功，将返回 TRUE。

MQLONG clusterWorkLoadMRU ();
返回最近使用的集群工作负载通道值，而不指示任何可能的错误。

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
提供集群工作负载使用队列值的副本。如果成功，将返回 TRUE。

MQLONG clusterWorkLoadUseQ ();
返回集群工作负载使用队列值，而不指示任何可能的错误。

ImqBoolean commandEvent (MQLONG & event);
提供命令事件值的副本。如果成功，将返回 TRUE。

MQLONG commandEvent ();
返回命令事件值，而不指示任何可能的错误。

ImqBoolean commandInputQueueName(ImqString & 名);
提供 命令输入队列名称 的副本。如果成功，将返回 TRUE。

ImqString commandInputQueueName();
返回 命令输入队列名称 ，而不指示任何可能的错误。

ImqBoolean commandLevel(MQLONG & 级别);
提供 命令级别 的副本。如果成功，将返回 TRUE。

MQLONG commandLevel();
返回 命令级别 ，而不指示任何可能的错误。

MQLONG commandServerControl ();
返回命令服务器启动值，而不指示任何可能的错误。

ImqBoolean commandServer 控制 (MQLONG 和 服务器);
提供命令服务器控制启动值的副本。如果成功，将返回 TRUE。

ImqBoolean commit ();
落实未落实的更改。如果成功，将返回 TRUE。

ImqBoolean connect ();

连接到具有给定 ImqObject 名称的队列管理器，缺省值为本地队列管理器。如果要连接到特定队列管理器，请在连接之前使用 ImqObject setName 方法。如果存在 通道引用，那么它用于将有关通道定义的信息传递到 MQCD 中的 MQCONN。MQCD 中的 ChannelType 设置为 MQCHT_CLNTCONN。对于服务器连接，将忽略仅对客户机连接有意义的 通道引用 信息。连接选项 会影响此方法的行为。如果成功，此方法会将 连接状态 设置为 TRUE。它返回新的连接状态。

如果存在第一个认证记录，那么将使用认证记录链来认证安全客户机通道的数字证书。

您可以将多个 ImqQueueManager 对象连接到同一队列管理器。所有用户都使用相同的 MQHCONN 连接句柄，并共享与线程关联的连接的 UOW 功能。要连接的第一个 ImqQueueManager 获取 MQHCONN 句柄。断开连接的最后一个 ImqQueue 管理器执行 MQDISC。

对于多线程程序，建议对每个线程使用单独的 ImqQueueManager 对象。

ImqBinary connectionId () 康斯特;

返回连接标识。

ImqBinary connectionTag () 康斯特;

返回 连接标记。

ImqBoolean setConnection 标记 (const MQBYTE128 标记 = 0);

设置 连接标记。如果标记为零，那么清除 连接标记。如果成功，此方法将返回 TRUE。

ImqBoolean setConnection 标记 (const ImqBinary 和 标记);

设置 连接标记。标记的 数据长度 必须为零 (以清除 连接标记) 或 MQ_CONN_TAG_LENGTH。如果成功，此方法将返回 TRUE。

MQLONG connectOptions() 康斯特;

返回 连接选项。

void setConnect 选项 (const MQLONG 选项 = MQCNO_NONE);

设置 连接选项。

ImqBoolean connectionStatus() 康斯特;

返回 连接状态。

ImqString cryptographicHardware ();

返回 加密硬件。

ImqBoolean setCryptographicHardware (const char * hardware = 0);

设置 加密硬件。如果成功，此方法将返回 TRUE。

ImqBoolean deadLetterQueueName(ImqString & 名);

提供 死信队列名称的副本。如果成功，将返回 TRUE。

ImqString deadLetterQueueName();

返回 dead-letter queue name 的副本，而不指示任何可能的错误。

ImqBoolean defaultTransmissionQueueName(ImqString & 名);

提供 缺省传输队列名称的副本。如果成功，将返回 TRUE。

ImqString defaultTransmissionQueueName();

返回 缺省传输队列名称，而不指示任何可能的错误。

ImqBoolean disconnect ();

断开与队列管理器的连接，并将 连接状态 设置为 FALSE。关闭与此对象关联的所有 ImqProcess 和 ImqQueue 对象，并在断开连接之前断开它们的 连接引用。如果有多个 ImqQueueManager 对象连接到同一队列管理器，那么只有最后一个断开连接的对象执行物理断开连接;其他对象执行逻辑断开连接。未落实的更改仅在物理断开连接时落实。

如果成功，此方法将返回 TRUE。如果在没有现有连接时调用，那么返回码也为 true。

ImqBoolean distributionLists(MQLONG & 支持);

提供 分发列表 值的副本。如果成功，将返回 TRUE。

MQLONG distributionLists();

返回 分布列表 值，而不指示任何可能的错误。

ImqBoolean dnsGroup (ImqString & group);

提供 DNS 组名的副本。如果成功，将返回 TRUE。

ImqString dnsGroup ();
返回 DNS 组名，而不指示可能的错误。

ImqBoolean dnsWlm (MQLONG & wlm);
提供 DNS WLM 值的副本。如果成功，将返回 TRUE。

MQLONG dnsWlm ();
返回 DNS WLM 值，而不指示任何可能的错误。

ImqAuthenticationRecord * firstAuthenticationRecord () 康斯特;
返回 第一个认证记录。

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * air = 0);
设置 第一个认证记录。

ImqObject * firstManagedObject () 康斯特;
返回 第一个受管对象。

ImqBoolean inhibitEvent(MQLONG & 事件);
提供 禁止事件的启用状态的副本。如果成功，将返回 TRUE。

MQLONG inhibitEvent();
返回 禁止事件 的启用状态，而不指示任何可能的错误。

ImqBoolean ipAddress 版本 (MQLONG 和版本);
提供 IP 地址版本值的副本。如果成功，将返回 TRUE。

MQLONG ipAddress 版本 ();
返回 IP 地址版本值，而不指示任何可能的错误。

ImqBoolean keepAlive (MQLONG & keepalive);
提供保持活动值的副本。如果成功，将返回 TRUE。

MQLONG keepAlive ();
返回保持活动值，而不指示任何可能的错误。

ImqString keyRepository ();
返回 密钥存储库。

ImqBoolean setKeyRepository (const char * repository = 0);
设置 密钥存储库。如果成功，将返回 TRUE。

ImqBoolean listenerTimer (MQLONG & timer);
提供侦听器计时器值的副本。如果成功，将返回 TRUE。

MQLONG listenerTimer ();
返回侦听器计时器值，而不指示任何可能的错误。

ImqBoolean localEvent(MQLONG & 事件);
提供 本地事件的启用状态的副本。如果成功，将返回 TRUE。

MQLONG localEvent();
返回 本地事件 的启用状态，而不指示任何可能的错误。

ImqBoolean loggerEvent (MQLONG & count);
提供记录器事件值的副本。如果成功，将返回 TRUE。

MQLONG loggerEvent ();
返回记录器事件值，而不指示任何可能的错误。

ImqBoolean luGroupName (ImqString & name);.
提供 LU 组名的副本。如果成功，将返回 TRUE

ImqString luGroup 名称 ();
返回 LU 组名而不指示任何可能的错误。

ImqBoolean lu62ARMSuffix (ImqString 和后缀);
提供 LU62 ARM 后缀的副本。如果成功，将返回 TRUE。

ImqString lu62ARMSuffix ();
返回 LU62 ARM 后缀而不指示任何可能的错误

ImqBoolean luName (ImqString & name);
提供 LU 名的副本。如果成功，将返回 TRUE。

ImqString luName ();
返回 LU 名而不指示任何可能的错误。

ImqBoolean maximumActive 通道 (MQLONG 和通道);
提供最大活动通道数值的副本。如果成功，将返回 TRUE。

MQLONG maximumActive 通道 ();
返回最大活动通道数值，而不指示任何可能的错误。

ImqBoolean maximumCurrent 通道 (MQLONG 和通道);
提供最大当前通道值的副本。如果成功，将返回 TRUE。

MQLONG maximumCurrent 通道 ();
返回最大当前通道值，而不指示任何可能的错误。

ImqBoolean maximumHandles(MQLONG & 数字);
提供最大句柄数的副本。如果成功，将返回 TRUE。

MQLONG maximumHandles();
返回最大句柄数，而不指示任何可能的错误。

ImqBoolean maximumLu62Channels (MQLONG 和通道);
提供最大 LU62 通道值的副本。如果成功，将返回 TRUE。

MQLONG maximumLu62Channels ();
返回最大 LU62 通道值，而不指示任何可能的错误。

ImqBoolean maximumMessageLength(MQLONG & 长度);
提供最大消息长度的副本。如果成功，将返回 TRUE。

MQLONG maximumMessage 长度 ();
返回最大消息长度，而不指示任何可能的错误。

ImqBoolean maximumPriority(MQLONG 和 *priority*);
提供最大优先级的副本。如果成功，将返回 TRUE。

MQLONG maximumPriority();
返回最大优先级的副本，而不指示任何可能的错误。

ImqBoolean maximumTcp 通道 (MQLONG 和通道);
提供最大 TCP 通道数值的副本。如果成功，将返回 TRUE。

MQLONG maximumTcp 通道 ();
返回最大 TCP 通道值，而不指示任何可能的错误。

ImqBoolean maximumUncommitted 消息 (MQLONG 和 *number*);
提供最大未落实消息数的副本。如果成功，将返回 TRUE。

MQLONG maximumUncommitted 消息 ();
返回最大未落实消息数，而不指示任何可能的错误。

ImqBoolean mqiAccounting (MQLONG 和 *statint*);
提供 MQI 记帐值的副本。如果成功，将返回 TRUE。

MQLONG mqiAccounting ();
返回 MQI 记帐值，而不指示任何可能的错误。

ImqBoolean mqiStatistics (MQLONG 和 *statmqi*);
提供 MQI 统计信息值的副本。如果成功，将返回 TRUE。

MQLONG mqiStatistics ();
返回 MQI 统计信息值，而不指示任何可能的错误。

ImqBoolean outboundPort 最大值 (MQLONG 和最大值);
提供最大出站端口值的副本。如果成功，将返回 TRUE。

MQLONG outboundPortMax ();
返回最大出站端口值，而不指示任何可能的错误。

ImqBoolean outboundPortMin (MQLONG & *min*);
提供最小出站端口值的副本。如果成功，将返回 TRUE。

MQLONG outboundPortMin ();
返回最小出站端口值，而不指示任何可能的错误。

ImqBinary 密码 () 康斯特;

返回客户机连接上使用的密码。

ImqBoolean setPassword (const ImqString & password);

设置客户机连接上使用的密码。

ImqBoolean setPassword (const char * = 0 password);

设置客户机连接上使用的密码。

ImqBoolean setPassword (const ImqBinary & password);

设置客户机连接上使用的密码。

ImqBoolean performanceEvent(MQLONG & 事件);

提供 性能事件 的启用状态的副本。如果成功，将返回 TRUE。

MQLONG performanceEvent();

返回 性能事件 的启用状态，而不指示任何可能的错误。

ImqBoolean platform(MQLONG & 平台);

提供 平台 的副本。如果成功，将返回 TRUE。

MQLONG 平台 ();

返回 平台 ，而不指示任何可能的错误。

ImqBoolean queueAccounting (MQLONG & acctq);

提供队列记帐值的副本。如果成功，将返回 TRUE。

MQLONG queueAccounting ();

返回队列记帐值，而不指示任何可能的错误。

ImqBoolean queueMonitoring (MQLONG & monq);

提供队列监视值的副本。如果成功，将返回 TRUE。

MQLONG queueMonitoring ();

返回队列监视值，而不指示任何可能的错误。

ImqBoolean queueStatistics (MQLONG 和 statq);

提供队列统计信息值的副本。如果成功，将返回 TRUE。

MQLONG queueStatistics ();

返回队列统计信息值，而不指示任何可能的错误。

ImqBoolean receiveTimeout (MQLONG & timeout);

提供接收超时值的副本。如果成功，将返回 TRUE。

MQLONG receiveTimeout ();

返回接收超时值，而不指示任何可能的错误。

ImqBoolean receiveTimeoutMin (MQLONG & min);

提供最小接收超时值的副本。如果成功，将返回 TRUE。

MQLONG receiveTimeoutMin ();

返回最小接收超时值，而不指示任何可能的错误。

ImqBoolean receiveTimeoutType (MQLONG & type);

提供接收超时类型的副本。如果成功，将返回 TRUE。

MQLONG receiveTimeout 类型 ();

返回接收超时类型，而不指示可能的错误。

ImqBoolean remoteEvent(MQLONG & 事件);

提供 远程事件 的启用状态的副本。如果成功，将返回 TRUE。

MQLONG remoteEvent();

返回 远程事件 的启用状态，而不指示任何可能的错误。

ImqBoolean repositoryName(ImqString & 名);

提供 存储库名称 的副本。如果成功，将返回 TRUE。

ImqString repositoryName();

返回 存储库名称 ，而不指示任何可能的错误。

ImqBoolean repositoryNameListName(ImqString & 名);

提供 存储库名称列表名称 的副本。如果成功，将返回 TRUE。

ImqString repositoryNameList 名称 ();
返回 **repository nameList name** 的副本，而不指示任何可能的错误。

ImqBoolean sharedQueueQueueManager 名称 (MQLONG 和名称);
提供共享队列管理器名称值的副本。如果成功，将返回 TRUE。

MQLONG sharedQueueQueueManager 名称 ();
返回共享队列管理器名称值，而不指示任何可能的错误。

ImqBoolean sslEvent (MQLONG & event);
提供 SSL 事件值的副本。如果成功，将返回 TRUE。

MQLONG sslEvent ();
返回 SSL 事件值，而不指示任何可能的错误。

ImqBoolean sslFips (MQLONG & sslfips);
提供 SSL FIPS 值的副本。如果成功，将返回 TRUE。

MQLONG sslFips ();
返回 SSL FIPS 值，而不指示任何可能的错误。

ImqBoolean sslKeyResetCount (MQLONG & count);
提供 SSL 密钥重置计数值的副本。如果成功，将返回 TRUE。

MQLONG sslKeyResetCount ();
返回 SSL 密钥重置计数值，而不指示任何可能的错误。

ImqBoolean startStopEvent(MQLONG & 事件);
提供 **启动-停止事件** 的启用状态的副本。如果成功，将返回 TRUE。

MQLONG startStop 事件 ();
返回 **start-stop event** 的启用状态，而不指示任何可能的错误。

ImqBoolean statisticsInterval (MQLONG 和 statint);
提供统计信息时间间隔值的副本。如果成功，将返回 TRUE。

MQLONG statisticsInterval ();
返回统计时间间隔值，而不指示任何可能的错误。

ImqBoolean syncPointAvailability(MQLONG & 同步);
提供 **syncpoint availability** 值的副本。如果成功，将返回 TRUE。

MQLONG syncPoint 可用性 ();
返回 **syncpoint availability** 值的副本，而不指示任何可能的错误。

ImqBoolean tcpName (ImqString & name);
提供 TCP 系统名称的副本。如果成功，将返回 TRUE。

ImqString tcpName ();
返回 TCP 系统名称，而不指示任何可能的错误。

ImqBoolean tcpStackType (MQLONG & type);
提供 TCP 堆栈类型的副本。如果成功，将返回 TRUE。

MQLONG tcpStack 类型 ();
返回 TCP 堆栈类型，而不指示任何可能的错误。

ImqBoolean traceRouteRecording (MQLONG & routerec);
提供跟踪路由记录值的副本。如果成功，将返回 TRUE。

MQLONG traceRoute 记录 ();
返回跟踪路由记录值，而不指示任何可能的错误。

ImqBoolean triggerInterval(MQLONG 和 interval);
提供 **触发器时间间隔** 的副本。如果成功，将返回 TRUE。

MQLONG triggerInterval();
返回 **触发器时间间隔**，而不指示任何可能的错误。

ImqBinary userId () 康斯特;
返回客户机连接上使用的用户标识。

ImqBoolean setUserId (const ImqString & id);
设置客户机连接上使用的用户标识。

ImqBoolean setUserId (const char * = 0 id);

设置客户机连接上使用的用户标识。

ImqBoolean setUserId (const ImqBinary & id);

设置客户机连接上使用的用户标识。

对象方法 (protected)

void setFirstManagedObject(const ImqObject * object = 0);

设置 第一个受管对象。

对象数据 (受保护)

MQHCONN 欧姆连接

WebSphere MQ 连接句柄 (仅当 连接状态 为 TRUE 时有意义)。

原因码

- MQRC_ATTRIBUTE_LOCKED
- MQRC_ENVIRONMENT_ERROR
- MQRC_FUNCTION_NOT_SUPPORTED
- MQRC_REFERENCE_ERROR
- (MQBACK 的原因码)
- (MQBEGIN 的原因码)
- (MQCMIT 的原因码)
- (MQCONN 的原因码)
- (MQCONNX 的原因码)
- (MQDISC 的原因码)
- (MQCONN 的原因码)

ImqReference 头 C++ 类

此类封装 MQRMH 数据结构的功能部件。

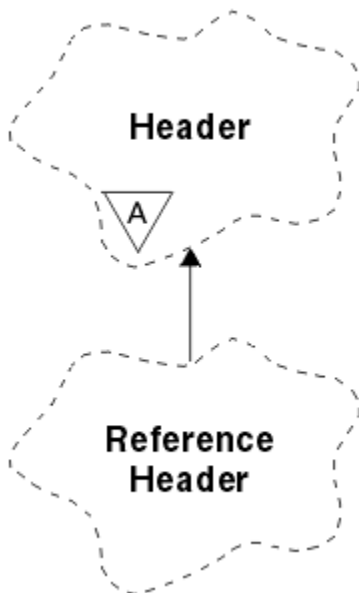


图 66: *ImqReference* 头类

此类与 [第 1190 页的『ImqReference 头交叉引用』](#) 中列出的 MQI 调用相关。

- [第 1267 页的『对象属性』](#)
- [第 1267 页的『构造函数』](#)
- [第 1267 页的『重载的 ImqItem 方法』](#)
- [第 1267 页的『对象方法 \(公用\)』](#)
- [第 1268 页的『对象数据 \(受保护\)』](#)
- [第 1268 页的『原因码』](#)

对象属性

目标环境

目标的环境。初始值为空字符串。

目标名称

数据目标的名称。初始值为空字符串。

实例标识

实例标识。长度为 MQ_OBJECT_INSTANCE_ID_LENGTH 的二进制值 (MQBYTE24)。初始值为 MQOII_NONE。

逻辑长度

此头后的消息数据的逻辑长度或预期长度。初始值为零。

逻辑偏移量

后续消息数据的逻辑偏移量，将在最终目标的整个数据上下文中进行解释。初始值为零。

逻辑偏移量 2

逻辑偏移量的高阶扩展。初始值为零。

引用类型

引用类型。初始值为空字符串。

源环境

源的环境。初始值为空字符串。

源名称

数据源的名称。初始值为空字符串。

构造函数

ImqReferenceHeader ();

缺省构造函数。

ImqReference 头 (const ImqReferenceHeader & 标题);

复制构造函数。

重载的 ImqItem 方法

虚拟 ImqBoolean copyOut(ImqMessage & 消息);

在开始时将 MQRMH 数据结构插入消息缓冲区，进一步移动现有消息数据，并将 *msg 格式* 设置为 MQFMT_REF_MSG_HEADER。

请参阅 [第 1216 页的『ImqHeader C++ 类』](#) 上的 ImqHeader 类方法描述以获取更多详细信息。

虚拟 ImqBoolean pasteIn(ImqMessage & 消息);

从消息缓冲区读取 MQRMH 数据结构。

要成功，ImqMessage 格式 必须为 MQFMT_REF_MSG_HEADER。

请参阅 [第 1216 页的『ImqHeader C++ 类』](#) 上的 ImqHeader 类方法描述以获取更多详细信息。

对象方法 (公用)

void 运算符 = (const ImqReferenceHeader & 标题);

从头复制实例数据，替换现有实例数据。

ImqString destinationEnvironment() const ;

返回 目标环境的副本。

void setDestinationEnvironment(const char * environment = 0);

设置 目标环境。

ImqString destinationName() const ;

返回 目标名称的副本。

void setDestinationName(const char * name = 0);

设置 目标名称。

ImqBinary instanceId() const ;

返回 实例标识的副本。

ImqBoolean setInstance 标识(const ImqBinary & 标识);

设置 实例标识。 *token* 的数据长度 必须为 0 或 MQ_OBJECT_INSTANCE_ID_LENGTH。 如果成功，此方法将返回 TRUE。

void setInstanceId(const MQBYTE24 id = 0);

设置 实例标识。 *id* 可以为零，这与指定 MQOII_NONE 相同。 如果 *id* 非零，那么它必须处理二进制数据的 MQ_OBJECT_INSTANCE_ID_LENGTH 字节。 使用预定义值 (例如 MQOII_NONE) 时，可能需要进行强制类型转换以确保签名匹配，例如 (MQBYTE *) MQOII_NONE。

MQLONG logicalLength() const ;

返回 逻辑长度。

void setLogicalLength(const MQLONG length);

设置 逻辑长度。

MQLONG logicalOffset() const ;

返回 逻辑偏移量。

void setLogicalOffset(const MQLONG offset);

设置 逻辑偏移量。

MQLONG logicalOffset2() const ;

返回 逻辑偏移量 2。

void setLogicalOffset2(const MQLONG offset);

设置 逻辑偏移量 2。

ImqString referenceType() const ;

返回 参考类型的副本。

void setReferenceType(const char * name = 0);

设置 引用类型。

ImqString sourceEnvironment() const ;

返回 源环境的副本。

void setSourceEnvironment(const char * environment = 0);

设置 源环境。

ImqString sourceName() const ;

返回 源名称的副本。

void setSourceName(const char * name = 0);

设置 源名称。

对象数据 (受保护)

MQRMH omqrmh

MQRMH 数据结构。

原因码

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_STRUC_LENGTH_ERROR

- MQRC_STRUC_ID_ERROR
- MQRC_IN 有数据
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR

ImqString C++ 类

此类为以 null 结束的字符串提供字符串存储和操作。

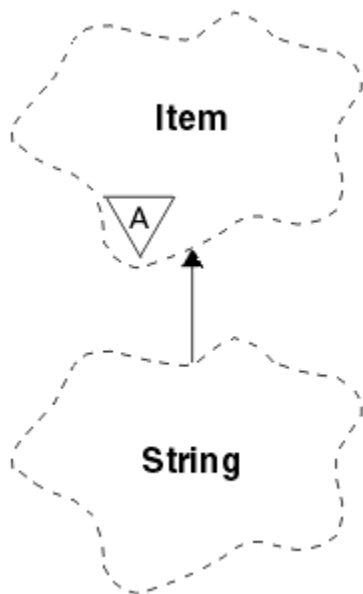


图 67: ImqString 类

在参数调用 **char *** 的大多数情况下，使用 ImqString 代替 **char ***。

- [第 1269 页的『对象属性』](#)
- [第 1269 页的『构造函数』](#)
- [第 1270 页的『类方法 \(public\)』](#)
- [第 1270 页的『重载的 ImqItem 方法』](#)
- [第 1270 页的『对象方法 \(公用\)』](#)
- [第 1273 页的『对象方法 \(protected\)』](#)
- [第 1273 页的『原因码』](#)

对象属性

个字符之后

存储器 中在尾部空字符之前的字符。

长度

字符中的字节数。如果没有 存储器，那么 **length** 为零。初始值为零。

存储器

任意大小的字节的易失性数组。在 存储器 中的 字符之后必须始终存在尾部空值，以便可以检测到 字符的结尾。方法确保保持此情况，但在直接设置数组中的字节时，确保修改后存在尾部空值。最初，没有 **storage** 属性。

构造函数

ImqString();

缺省构造函数。

ImqString(const ImqString & 字符串);

复制构造函数。

ImqString(const char c);

字符由 *c* 组成。

ImqString(const char * text);

字符是从 *text* 复制的。

ImqString(const void * buffer, const size_t length);

从 *buffer* 开始复制 *length* 个字节，并将其分配给 **字符**。将对复制的任何空字符进行替换。替换字符是句点 (.)。没有特别考虑任何其他不可打印或不可显示的字符被复制。

类方法 (public)

静态 ImqBoolean 副本 (char * destination-buffer, const size_t length, const char * source-buffer, const char pad = 0);

将最多 *length* 个字节从 *source-buffer* 复制到 *destination-buffer*。如果 *source-buffer* 中的字符数不足，请使用 *pad* 字符填充 *destination-buffer* 中的剩余空间。*source-buffer* 可以为零。如果 *length* 也为零，那么 *destination-buffer* 可以为零。任何错误代码都将丢失。如果成功，此方法将返回 TRUE。

static ImqBoolean copy (char * 目标缓冲区, const size_t 长度, const char * 源缓冲区, ImqError & 错误对象, const char 垫 = 0);

将最多 *length* 个字节从 *source-buffer* 复制到 *destination-buffer*。如果 *source-buffer* 中的字符数不足，请使用 *pad* 字符填充 *destination-buffer* 中的剩余空间。*source-buffer* 可以为零。如果 *length* 也为零，那么 *destination-buffer* 可以为零。任何错误代码都在 *error-object* 中设置。如果成功，此方法将返回 TRUE。

重载的 ImqItem 方法

虚拟 ImqBoolean copyOut(ImqMessage & 消息);

将 **字符** 复制到消息缓冲区，以替换任何现有内容。将 *msg 格式* 设置为 MQFMT_STRING。

请参阅父类方法描述以获取更多详细信息。

虚拟 ImqBoolean pasteIn(ImqMessage & 消息);

通过从消息缓冲区传输剩余数据 (替换现有 **字符**) 来设置 **字符**。

要成功，*msg* 对象的 **encoding** 必须是 MQENC_NATIVE。使用 MQGMO_CONVERT 检索到 MQENC_NATIVE 的消息。

要成功，*ImqMessage 格式* 必须是 MQFMT_STRING。

请参阅父类方法描述以获取更多详细信息。

对象方法 (公用)

char & operator [] (const size_t 偏移量) const ;

在 **存储器** 中引用偏移量为 *offset* 的字符。请确保相关字节存在且可寻址。

ImqString 运算符 () (const size_t offset, const size_t length = 1) 康斯特;

通过从 *offset* 开始从 **characters** 复制字节来返回子串。如果 *length* 为零，那么返回其余 **字符**。如果 *offset* 和 *length* 的组合未在 **字符** 中生成引用，那么返回空的 **ImqString**。

void operator = (const ImqString & 字符串);

从 *string* 复制实例数据，替换现有实例数据。

ImqString 运算符 + (const char c) 康斯特;

返回将 *c* 附加到 **字符** 的结果。

ImqString 运算符 + (const char * text) 康斯特;

返回将 *text* 附加到 **字符** 的结果。这也可以倒过来。例如：

```
strOne + "string two" ;  
"string one" + strTwo ;
```

注: 虽然大多数编译器接受 `strOne + "string two"`; Microsoft Visual C++ 需要 `strOne + (char *) "string two"`;

ImqString operator + (const ImqString & string1) const ;

返回将 `string1` 附加到 字符的结果。

ImqString 运算符 + (const double number) 康斯特;

返回在转换为文本后将 `number` 附加到 字符的结果。

ImqString 运算符 + (const long number) 康斯特;

返回在转换为文本后将 `number` 附加到 字符的结果。

void 运算符 += (const char c);

将 `c` 附加到 字符。

void 运算符 += (const char * text);

将文本 追加到 字符。

void operator += (const ImqString & 字符串);

将 `string` 附加到 字符。

void 运算符 += (const double number);

在转换为文本之后, 将 `number` 附加到 字符。

void 运算符 += (const long number);

在转换为文本之后, 将 `number` 附加到 字符。

运算符 char * () 康斯特;

返回 `storage` 中第一个字节的地址。此值可以为零, 并且具有易失性。仅将此方法用于只读目的。

ImqBoolean operator < (const ImqString & 字符串) const ;

使用 `Compare` 方法将 字符 与 `string` 的字符进行比较。如果小于或等于, 那么结果为 TRUE; 如果大于或等于, 那么结果为 FALSE。

ImqBoolean operator > (const ImqString & 字符串) const ;

使用 `Compare` 方法将 字符 与 `string` 的字符进行比较。如果大于或等于 FALSE, 那么结果为 TRUE。

ImqBoolean operator <= (const ImqString & 字符串) const ;

使用 `Compare` 方法将 字符 与 `string` 的字符进行比较。如果小于或等于, 那么结果为 TRUE; 如果大于, 那么结果为 FALSE。

ImqBoolean operator >= (const ImqString & 字符串) const ;

使用 `Compare` 方法将 字符 与 `string` 的字符进行比较。如果大于或等于, 那么结果为 TRUE; 如果小于, 那么结果为 FALSE。

ImqBoolean operator == (const ImqString & 字符串) const ;

使用 `Compare` 方法将 字符 与 `string` 的字符进行比较。它返回 TRUE 或 FALSE。

ImqBoolean operator != (const ImqString & 字符串) const ;

使用 `Compare` 方法将 字符 与 `string` 的字符进行比较。它返回 TRUE 或 FALSE。

short compare(const ImqString & 字符串) const ;

将 字符 与 `string` 的字符进行比较。如果 字符 相等, 那么结果为零, 如果小于, 那么结果为负, 如果大于, 那么结果为正。比较区分大小写。将空 `ImqString` 视为小于非空 `ImqString`。

ImqBoolean copyOut(char * buffer, const size_t length, const char pad = 0);

将最多 `length` 个字节从 字符 复制到缓冲区。如果 字符数 不足, 请使用 `pad` 字符填充 `buffer` 中的剩余空间。如果 `length` 也为零, 那么 `buffer` 可以为零。如果成功, 将返回 TRUE。

size_t copyOut(long & 数字) const ;

从文本转换后从 字符 设置 `number`, 并返回转换中涉及的字符数。如果此值为零, 那么表示未执行任何转换, 并且未设置 `number`。可转换字符序列必须以以下值开头:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString & 令牌, const char C = ' ') const ;

如果 **字符** 包含一个或多个不同于 *c* 的字符，请将标记标识为此类字符的第一个连续序列。在这种情况下，*token* 设置为该序列，返回的值是前导字符数 *c* 与序列中字节数的总和。否则，返回零，并且不设置 *token*。

size_t cutOut(long & 数字);

设置 *number* 作为 **copy** 方法，但也从 **字符** 中除去由返回值指示的字节数。例如，可以使用 **cutOut(number)** 将以下示例中显示的字符串分割为三个数字 三次：

```
strNumbers = "-1 0      +55 "
while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString & 令牌, const char C = ' ')

将 *token* 设置为 **copyOut** 方法，并从 **字符** 中除去 *strToken* 字符以及 *token* 字符之前的任何字符 *c*。如果 *c* 不是空白，那么除去直接继承 *token* 字符的字符 *c*。返回除去的字符数。例如，可以使用 **cutOut(token)** 将以下示例中显示的字符串分割为三个标记 三次：

```
strText = " Program Version 1.1 "
while ( strText.cutOut( token ) );
// token becomes "Program", then "Version",
// then "1.1" leaving strText == " "
```

以下示例显示如何解析 DOS 路径名：

```
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 & 字符串);

在 **字符** 中的任何位置搜索 *string* 的完全匹配项。如果找不到匹配项，那么将返回 FALSE。否则，将返回 TRUE。如果 *string* 为空，那么返回 TRUE。

ImqBoolean find(const ImqString & 字符串, size_t & 偏移量);

从偏移量 *offset* 开始，在 **字符** 内的某个位置搜索 *string* 的完全匹配项。如果 *string* 为空，那么它将返回 TRUE 而不更新 *offset*。如果找不到匹配项，那么它将返回 FALSE (*offset* 的值可能已增大)。如果找到匹配项，那么它将返回 TRUE，并将 *offset* 更新为 **字符** 中 *string* 的偏移量。

size_t length () 康斯特;

返回 *length*。

ImqBoolean pasteIn(const double number, const char * format = "%f");

在转换为文本之后，将 *number* 附加到 **字符**。如果成功，将返回 TRUE。

规范 *format* 用于格式化浮点转换。如果指定了此参数，那么它必须是适合与 **printf** 和浮点数一起使用的值，例如 **%3f**。

ImqBoolean pasteIn(const long number);

在转换为文本之后，将 *number* 附加到 **字符**。如果成功，将返回 TRUE。

ImqBoolean pasteIn(const void * buffer, const size_t length);

将 *length* 个字节从缓冲区追加到 **字符**，并添加最终尾部空值。替换复制的任何空字符。替换字符是句点 (.)。没有特别考虑任何其他不可打印或不可显示的字符被复制。如果成功，此方法将返回 TRUE。

ImqBoolean 集 (const char * *buffer*, const size_t *length*);

设置固定长度字符字段中的 **字符**，该字段可能包含空值。如果需要，将 null 附加到固定长度字段中的字符。如果成功，此方法将返回 TRUE。

ImqBoolean setStorage(const size_t *length*);

分配 (或重新分配) **存储器**。保留任何原始 **字符**，包括任何尾部空字符 (如果仍有这些字符的空间)，但不初始化任何其他存储器。

如果成功，此方法将返回 TRUE。

size_t storage () 康斯特;

返回 **storage** 中的字节数。

size_t stripLeading(const char *c* = " ");

从 **字符** 中除去前导字符 *c*，并返回除去的数字。

size_t stripTrailing(const char *c* = " ");

从 **字符** 中删除尾部字符 *c*，并返回除去的数字。

ImqString upperCase() 康斯特;

返回 **字符** 的大写副本。

对象方法 (protected)

ImqBoolean 分配 (const ImqString & 字符串);

等价于等效的 **运算符 =** 方法，但非虚拟方法。如果成功，将返回 TRUE。

原因码

- MQRC_DATA_截断
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_BUFFER_ERROR
- MQRC_INCONSISTENT_FORMAT

ImqTrigger C++ 类

此类封装 MQTM (触发器消息) 数据结构。

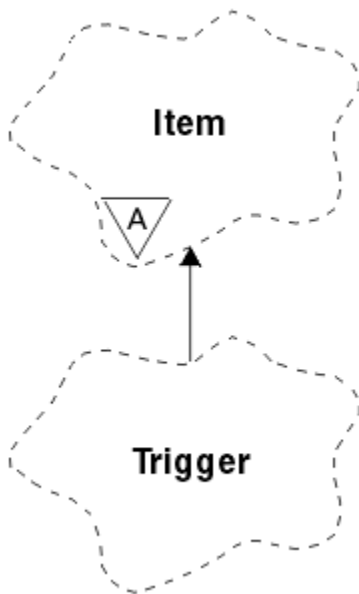


图 68: *ImqTrigger* 类

此类的对象通常由触发器监视器程序使用。触发器监视器程序的任务是等待这些特定消息并对其执行操作，以确保在消息等待时启动其他 WebSphere MQ 应用程序。

请参阅 IMQSTRG 样本程序以获取用法示例。

- [第 1274 页的『对象属性』](#)
- [第 1274 页的『构造函数』](#)
- [第 1275 页的『重载的 ImqItem 方法』](#)
- [第 1275 页的『对象方法 \(公用\)』](#)
- [第 1275 页的『对象数据 \(受保护\)』](#)
- [第 1276 页的『原因码』](#)

对象属性

应用程序标识

发送消息的应用程序的身份。初始值为空字符串。

应用程序类型

发送消息的应用程序的类型。初始值为零。可以使用以下附加值：

- 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

环境数据

进程的环境数据。初始值为空字符串。

流程名称

进程名称。初始值为空字符串。

队列名称

要启动的队列的名称。初始值为空字符串。

触发器数据

触发进程的数据。初始值为空字符串。

用户数据

进程的用户数据。初始值为空字符串。

构造函数

ImqTrigger();

缺省构造函数。

ImqTrigger(const ImqTrigger & 触发);

复制构造函数。

重载的 ImqItem 方法

虚拟 ImqBoolean copyOut(ImqMessage & 消息);

将 MQTM 数据结构写入消息缓冲区, 以替换任何现有内容。将 *msg* 格式设置为 MQFMT_TRIGGER。

请参阅位于第 1220 页的『ImqItem C++ 类』的 ImqItem 类方法描述以获取更多详细信息。

虚拟 ImqBoolean pasteIn(ImqMessage & 消息);

从消息缓冲区读取 MQTM 数据结构。

要成功, ImqMessage 格式 必须为 MQFMT_TRIGGER。

请参阅位于第 1220 页的『ImqItem C++ 类』的 ImqItem 类方法描述以获取更多详细信息。

对象方法 (公用)

void 运算符 = (const ImqTrigger & 触发);

从触发器复制实例数据, 以替换现有实例数据。

ImqString applicationId() const ;

返回 应用程序标识的副本。

void setApplicationId(const char * id);

设置 应用程序标识。

MQLONG applicationType() const ;

返回 应用程序类型。

void setApplicationType(const MQLONG type);

设置 应用程序类型。

ImqBoolean copyOut(MQTMC2 * ptmc2);

封装 MQTM 数据结构, 即在启动队列上接收到的 MQTM 数据结构。填写由调用者提供的等效 MQTMC2 数据结构, 并将 QMgrName 字段 (在 MQTM 数据结构中不存在) 设置为所有空白。MQTMC2 数据结构传统上用作触发器监视器启动的应用程序的参数。如果成功, 此方法将返回 TRUE。

ImqString environmentData() const ;

返回 环境数据的副本。

void setEnvironmentData(const char * data);

设置 环境数据。

ImqString processName() const ;

返回 流程名称的副本。

void setProcessName(const char * name);

将用空格填充的 进程名称 设置为 48 个字符。

ImqString queueName() const ;

返回 队列名称的副本。

void setQueueName(const char * name);

设置 队列名称, 用空格填充到 48 个字符。

ImqString triggerData() const ;

返回 触发器数据的副本。

void setTriggerData(const char * data);

设置 触发器数据。

ImqString userData() const ;

返回 用户数据的副本。

void setUserData(const char * data);

设置 用户数据。

对象数据 (受保护)

MQTM omqtm

MQTM 数据结构。

原因码

- MQRC_NULL_POINTER
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

ImqWork 头 C++ 类

此类封装 MQWIH 数据结构的特定功能。

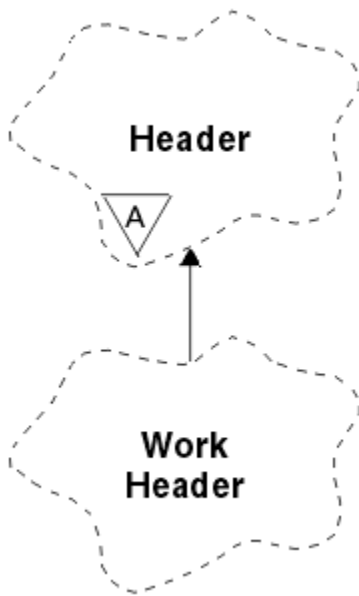


图 69: ImqWork 头类

此类的对象由将消息放入由 z/OS 工作负载管理器管理的队列的应用程序使用。

- [第 1276 页的『对象属性』](#)
- [第 1276 页的『构造函数』](#)
- [第 1277 页的『重载的 ImqItem 方法』](#)
- [第 1277 页的『对象方法 \(公用\)』](#)
- [第 1277 页的『对象数据 \(受保护\)』](#)
- [第 1277 页的『原因码』](#)

对象属性

消息标记 (message token)

z/OS 工作负载管理器的消息令牌，长度为 MQ_MSG_TOKEN_LENGTH。初始值为 MQMTOK_NONE。

服务名称

进程的 32 个字符的名称。该名称最初为空白。

服务步骤

进程中步骤的 8 字符名称。该名称最初为空白。

构造函数

ImqWorkHeader ();

缺省构造函数。

ImqWorkHeader(const ImqWorkHeader & 标题);

复制构造函数。

重载的 ImqItem 方法

virtual ImqBoolean copyOut(ImqMessage & 消息);

将 MQWIH 数据结构插入到消息缓冲区的开头，进一步移动现有消息数据，并将 *msg* 格式设置为 MQFMT_WORK_INFO_HEADER。

请参阅父类方法描述以获取更多详细信息。

virtual ImqBoolean pasteIn(ImqMessage & 消息);

从消息缓冲区读取 MQWIH 数据结构。

要成功，*msg* 对象的编码必须是 MQENC_NATIVE。使用 MQGMO_CONVERT 检索到 MQENC_NATIVE 的消息。

ImqMessage 格式必须为 MQFMT_WORK_INFO_HEADER。

请参阅父类方法描述以获取更多详细信息。

对象方法 (公用)

void operator = (const ImqWorkHeader & 标题);

从头复制实例数据，替换现有实例数据。

ImqBinary messageToken () 康斯特;

返回 消息令牌。

ImqBoolean setMessageToken(const ImqBinary & 令牌);

设置 消息令牌。*token* 的数据长度必须为零或 MQ_MSG_TOKEN_LENGTH。如果成功，将返回 TRUE。

void setMessageToken (const MQBYTE16 token = 0);

设置 消息令牌。*token* 可以为零，这与指定 MQMTOK_NONE 相同。如果 *token* 非零，那么它必须寻址二进制数据的 MQ_MSG_TOKEN_LENGTH 字节。

使用预定义值 (例如 MQMTOK_NONE) 时，您可能需要进行强制类型转换以确保签名匹配; 例如 (MQBYTE *) MQMTOK_NONE。

ImqString serviceName () 康斯特;

返回 服务名称，包括尾部空格。

void setServiceName (const char * name);

设置 服务名称。

ImqString serviceStep () 康斯特;

返回 服务步骤，包括尾部空格。

void setService 步骤 (const char * 步骤);

设置 服务步骤。

对象数据 (受保护)

MQWIH omqwih

MQWIH 数据结构。

原因码

- MQRC_BINARY_DATA_LENGTH_ERROR

Java 库的 IBM WebSphere MQ 类

Java 库的 IBM WebSphere MQ 类的位置因平台而异。您启动应用程序时需指定此位置。

要指定 Java 本机接口 (JNI) 库的位置, 请使用以下格式的 **java** 命令启动应用程序:

```
java -Djava.library.path=library_path application_name
```

其中 *library_path* 是 WebSphere MQ Java 库类 (包括 JNI 库) 的路径。第 1278 页的表 615 显示每个平台的 WebSphere MQ Java 库类的位置。

| 平台 | 包含 WebSphere MQ Java 库类的目录 |
|--|--|
| AIX | <i>MQ_INSTALLATION_PATH</i> /java/lib (32 位库) <i>MQ_INSTALLATION_PATH</i> /java/lib64 (64 位库) |
| HP-UX Linux (POWER, x86-64 和 zSeries s390x 平台) Solaris (x86-64 和 SPARC 平台) | <i>MQ_INSTALLATION_PATH</i> /java/lib (32 位库) <i>MQ_INSTALLATION_PATH</i> /java/lib64 (64 位库) |
| Linux (x86 平台) | <i>MQ_INSTALLATION_PATH</i> /java/lib |
| Windows | <i>MQ_INSTALLATION_PATH</i> \java\lib (32 位库) <i>MQ_INSTALLATION_PATH</i> \java\lib64 (64 位库) |

MQ_INSTALLATION_PATH 表示安装 WebSphere MQ 的高级目录。

注:

1. 在 AIX, HP-UX, Linux (Power 平台) 或 Solaris 上, 使用 32 位库或 64 位库。仅当在 64 位平台上的 64 位 Java 虚拟机 (JVM) 中运行应用程序时, 才使用 64 位库。否则, 请使用 32 位库。
2. 在 Windows 上, 可以使用 PATH 环境变量来指定 WebSphere MQ classes for Java 库的位置, 而不是在 **java** 命令上指定它们的位置。
3. 要在 IBM i 上以绑定方式使用 WebSphere MQ Java 类, 请确保库 QMQMJAVA 位于库列表中。

相关任务

[使用 WebSphere MQ classes for Java](#)

IBM WebSphere MQ classes for JMS 对象的属性

IBM WebSphere MQ classes for JMS 中的所有对象都具有属性。不同的属性适用于不同的对象类型。不同的属性具有不同的允许值, 并且管理工具和程序代码之间的符号属性值不同。

IBM WebSphere MQ classes for JMS 提供了一些工具, 用于使用 WebSphere MQ JMS 管理工具 WebSphere MQ Explorer 或在应用程序中设置和查询对象的属性。许多属性仅与对象类型的特定子集相关。

有关如何使用 WebSphere MQ JMS 管理工具的信息, 请参阅 [使用 WebSphere MQ JMS 管理工具](#)。

第 1279 页的表 616 提供每个属性的简要描述, 并针对每个属性显示其应用的对象类型。使用关键字标识对象类型; 请参阅 [JMS 对象类型](#) 以获取这些对象类型的说明。

数字是指表末尾的注释。另请参阅第 1326 页的『[WebSphere MQ JMS 对象类的属性之间的依赖关系](#)』。

属性由以下格式的 "名称/值" 对组成:

```
PROPERTY_NAME(property_value)
```

此部分列表中的主题针对每个属性，属性的名称和简要描述，并显示在管理工具中使用的有效属性值。以及用于在应用程序中设置属性值的 `set` 方法。这些主题还显示每个属性的有效属性值以及工具中使用的符号属性值与其可编程等效项之间的映射。

属性名称不区分大小写，并且仅限于这些主题中显示的一组可识别的名称。

| 属性 | 缩写 | 对象类型 | | | | | | | |
|---|---------|----------------|-----|----------------|---|---|------|-------|-------|
| | | CF | QCF | TCF | Q | T | XACF | XAQCF | XATCF |
| 第 1282 页的『APPLICATIONNAME』 | APPNAME | Y | Y | Y | | | Y | Y | Y |
| 第 1282 页的『ASYNCEXCEPTION』 | AEX | Y | Y | Y | | | Y | Y | Y |
| 第 1283 页的『BROKERCCDURSUBQ』 ¹ | CCDSUB | | | | | Y | | | |
| 第 1284 页的『BROKERCCSUBQ』 ¹ | CCSUB | Y | | Y | | | Y | | Y |
| 第 1284 页的『BROKERCONQ』 ¹ | BCON | Y | | Y | | | Y | | Y |
| 第 1285 页的『BROKERDURSUBQ』 ¹ | BDSUB | | | | | Y | | | |
| 第 1285 页的『BROKERPUBQ』 ¹ | BPUB | Y | | Y | | Y | Y | | Y |
| 第 1285 页的『BROKERPUBQMGR』 ¹ | BPQM | | | | | Y | | | |
| 第 1286 页的『BROKERQMGR』 ¹ | BQM | Y | | Y | | | Y | | Y |
| 第 1286 页的『BROKERSUBQ』 ¹ | BSUB | Y | | Y | | | Y | | Y |
| 第 1287 页的『BROKERVER』 ¹ | BVER | 是 ² | | 是 ² | | Y | Y | | Y |
| 第 1287 页的『CCDTURL』 ³ | CCDT | Y | Y | Y | | | Y | Y | Y |
| 第 1288 页的『CCSID』 | CCS | Y | Y | Y | Y | Y | Y | Y | Y |
| 第 1288 页的『通道』 ³ | CHAN | Y | Y | Y | | | Y | Y | Y |
| 第 1289 页的『CLEANUP』 ¹ | CL | Y | | Y | | | Y | | Y |
| 第 1289 页的『CLEANUPINT』 ¹ | CLINT | Y | | Y | | | Y | | Y |
| 第 1290 页的『connectionNameList』 | 国家名单 | Y | Y | Y | | | | | |
| 第 1290 页的『CLIENTRECONNECTOPTIONS』 | CROPT | Y | Y | Y | | | | | |
| 第 1291 页的『CLIENTRECONNECTTIMEOUT』 | CRT | Y | Y | Y | | | | | |
| 第 1291 页的『CLIENTID』 | CID | 是 ² | Y | 是 ² | | | Y | Y | Y |
| 第 1292 页的『CLONESUPP』 | CLS | Y | | Y | | | Y | | Y |
| 第 1292 页的『COMPHDR』 | HC | Y | | Y | | | Y | | Y |
| 第 1293 页的『COMPMSG』 | MC | Y | Y | Y | | | Y | Y | Y |
| 第 1293 页的『CONNOPT』 | CNOPT | Y | Y | Y | | | Y | Y | Y |
| 第 1294 页的『CONNTAG』 | CNTAG | Y | Y | Y | | | Y | Y | Y |
| 第 1294 页的『DESCRIPTION』 | DESC | 是 ² | Y | 是 ² | Y | Y | Y | Y | Y |
| 第 1295 页的『DIRECTAUTH』 | DAUTH | 是 ² | | 是 ² | | | | | |
| 第 1295 页的『ENCODING』 | ENC | | | | Y | Y | | | |

表 616: 属性名和适用的对象类型 (继续)

| 属性 | 缩写 | 对象类型 | | | | | | | | |
|---|---------|----------------|-----|----------------|---|---|------|-------|-------|--|
| | | CF | QCF | TCF | Q | T | XACF | XAQCF | XATCF | |
| 第 1296 页的『EXPIRY』 | EXP | | | | Y | Y | | | | |
| 第 1297 页的『FAILIFQUIESCE』 | FIQ | Y | Y | Y | Y | Y | Y | Y | Y | |
| 第 1297 页的『HOSTNAME』 | HOST | 是 ² | Y | 是 ² | | | Y | Y | Y | |
| 第 1298 页的『LOCALADDRESS』 | LA | 是 ² | Y | 是 ² | | | Y | Y | Y | |
| 第 1298 页的『MAPNAMESTYLE』 | MNST | Y | Y | Y | | | Y | Y | Y | |
| 第 1299 页的『MAXBUFFSIZE』 | MBSZ | 是 ² | | 是 ² | | | | | | |
| 第 1299 页的『MDREAD』 | MDR | | | | Y | Y | | | | |
| 第 1300 页的『MDWRITE』 | MDW | | | | Y | Y | | | | |
| 第 1300 页的『MDMSGCTX』 | MDCTX | | | | Y | Y | | | | |
| 第 1301 页的『MSGBATCHSZ』 ¹ | MBS | Y | Y | Y | | | Y | Y | Y | |
| 第 1301 页的『MSGBODY』 | MBODY | | | | Y | Y | | | | |
| 第 1302 页的『MSGRETENTION』 | MRET | Y | Y | | | | Y | Y | | |
| 第 1302 页的『MSGSELECTION』 ¹ | MSEL | Y | | Y | | | Y | | Y | |
| 第 1303 页的『MULTICAST』 | MCAST | 是 ² | | 是 ² | | Y | | | | |
| 第 1303 页的 『OPTIMISTICPUBLICATION』 ¹ | OPTPUB | Y | | Y | | | | | | |
| 第 1304 页的 『OUTCOMENOTIFICATION』 ¹ | NOTIFY | Y | | Y | | | | | | |
| 第 1304 页的『PERSISTENCE』 | PER | | | | Y | Y | | | | |
| 第 1305 页的『POLLINGINT』 ¹ | PINT | Y | Y | Y | | | Y | Y | Y | |
| 第 1306 页的『端口』 | 端口 | 是 ² | Y | 是 ² | | | Y | Y | Y | |
| 第 1306 页的『PRIORITY』 | PRI | | | | Y | Y | | | | |
| 第 1307 页的『PROCESSDURATION』 ¹ | PROCDUR | Y | | Y | | | | | | |
| 第 1307 页的『PROVIDERVERSION』 | PVER | Y | Y | Y | | | Y | Y | Y | |
| 第 1308 页的『PROXYHOSTNAME』 | PHOST | 是 ² | | 是 ² | | | | | | |
| 第 1309 页的『PROXYPORT』 | PPORT | 是 ² | | 是 ² | | | | | | |
| 第 1309 页的『PUBACKINT』 ¹ | PAI | Y | | Y | | | Y | | Y | |
| 第 1309 页的『PUTASYNCALLOWED』 | PAALD | | | | Y | Y | | | | |
| 第 1310 页的『QMANAGER』 | QMGR | Y | Y | Y | Y | | Y | Y | Y | |
| 第 1310 页的『队列』 | QU | | | | Y | | | | | |
| 第 1311 页的 『READAHEADALLOWED』 | 后勤司 | | | | Y | Y | | | | |
| 第 1311 页的 『READAHEADCLOSEPOLICY』 | RACP | | | | Y | Y | | | | |

表 616: 属性名和适用的对象类型 (继续)

| 属性 | 缩写 | 对象类型 | | | | | | | |
|---|---------|----------------|-----|----------------|---|---|------|-------|-------|
| | | CF | QCF | TCF | Q | T | XACF | XAQCF | XATCF |
| 第 1312 页的『RECEIVECCSID』 | RCCS | | | | Y | Y | | | |
| 第 1312 页的 『RECEIVECONVERSION』 | RCNV | | | | Y | Y | | | |
| 第 1313 页的『RECEIVEISOLATION』 ¹ | RCVISOL | Y | | Y | | | | | |
| 第 1313 页的『RECEXIT』 | RCX | Y | Y | Y | | | Y | Y | Y |
| 第 1314 页的『RECEXITINIT』 | RCXI | Y | Y | Y | | | Y | Y | Y |
| 第 1314 页的『REPLYTOSTYLE』 | RTOST | | | | Y | Y | | | |
| 第 1315 页的『RESCANINT』 ¹ | RINT | Y | Y | | | | Y | Y | |
| 第 1315 页的『SECEXIT』 | SCX | Y | Y | Y | | | Y | Y | Y |
| 第 1316 页的『SECEXITINIT』 | SCXI | Y | Y | Y | | | Y | Y | Y |
| 第 1316 页的『SENDCHECKCOUNT』 | SCC | Y | Y | Y | | | Y | Y | Y |
| 第 1317 页的『SENDEXIT』 | SDX | Y | Y | Y | | | Y | Y | Y |
| 第 1317 页的『SENDEXITINIT』 | SDXI | Y | Y | Y | | | Y | Y | Y |
| 第 1318 页的 『SHARECONVALLOWED』 | 后勤司 | Y | Y | Y | | | Y | Y | Y |
| 第 1318 页的『SPARSESUBS』 ¹ | SSUBS | Y | | Y | | | | | |
| 第 1319 页的『SSLCIPHERSUITE』 | SCPHS | Y | Y | Y | | | Y | Y | Y |
| 第 1319 页的『SSLCLRL』 | SCRL | Y | Y | Y | | | Y | Y | Y |
| 第 1319 页的『SSLFIPSREQUIRED』 | SFIPS | Y | Y | Y | | | Y | Y | Y |
| 第 1320 页的『SSLPEERNAME』 | SPEER | Y | Y | Y | | | Y | Y | Y |
| 第 1320 页的『SSLRESETCOUNT』 | SRC | Y | Y | Y | | | Y | Y | Y |
| 第 1321 页的『STATREFRESHINT』 ¹ | SRI | Y | | Y | | | Y | | Y |
| 第 1321 页的『SUBSTORE』 ¹ | SS | Y | | Y | | | Y | | Y |
| 第 1322 页的 『SYNCPOINTALLGETS』 | SPAG | Y | Y | Y | | | Y | Y | Y |
| 第 1322 页的『TARGCLIENT』 | TC | | | | Y | Y | | | |
| 第 1323 页的 『TARGCLIENTMATCHING』 | TCM | Y | Y | | | | Y | Y | |
| 第 1323 页的『TEMPMODEL』 | TM | Y | Y | | | | Y | Y | |
| 第 1324 页的『TEMPQPREFIX』 | TQP | Y | Y | | | | Y | Y | |
| 第 1324 页的『TEMPTOPICPREFIX』 | TTP | Y | | Y | | | Y | | Y |
| 第 1324 页的『TOPIC』 | TOP | | | | | Y | | | |
| 第 1325 页的『TRANSPORT』 | TRAN | 是 ² | Y | 是 ² | | | Y | Y | Y |
| 第 1325 页的『WILDCARDFORMAT』 | WCFMT | Y | | Y | | | Y | | Y |

表 616: 属性名和适用的对象类型 (继续)

| 属性 | 缩写 | 对象类型 | | | | | | | |
|---|----|------|-----|-----|---|---|------|-------|-------|
| | | CF | QCF | TCF | Q | T | XACF | XAQCF | XATCF |
| 注: 1. 此属性可以与 WebSphere MQ classes for JMS 的版本 7.0 配合使用, 但对于连接到版本 7.0 队列管理器的应用程序不起作用, 除非连接工厂的 PROVIDERVERSION 属性设置为小于 7 的版本号。 2. 当使用与代理的实时连接时, ConnectionFactory 或 TopicConnectionFactory 对象仅支持 BROKERVER, CLIENTID, DESCRIPTION, DIRECTAUTH, HOSTNAME, LOCALADDRESS, MAXBUFFSIZE, 多点广播, PORT, PROXYHOSTNAME, PROXYPORT 和属性。 3. 不能同时设置对象的 CCDURL 和 CHANNEL 属性。 | | | | | | | | | |

APPLICATIONNAME

应用程序可以设置一个名称来确定与队列管理器的连接。此应用程序名称由 **DISPLAY CONN MQSC/PCF** 命令 (其中该字段称为 **APPLTAG**) 或 IBM WebSphere MQ 资源管理器 **应用程序连接** 屏幕 (其中该字段称为 **App name**) 显示。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :APPLICATIONNAME

JMS 管理工具短名称 :APPNAME

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setApp 名称 ()
- MQConnectionFactory.getApp 名称 ()

值

任何长度不超过 28 个字符的有效字符串。如果需要, 将通过除去前导包名来调整较长的名称。例如, 如果调用类是 com.example.MainApp, 将使用全名, 但如果调用类是 com.example.dictionaryAndThesaurus.multilingual.mainApp, 将使用名称 multilingual.mainApp, 因为这是适合可用长度要求的类名称和最右侧包名称的最长组合。

如果类名称本身长度超过 28 个字符, 就会被截断。例如, com.example.mainApplicationForSecondTestCase 会被截断为 mainApplicationForSecondTest。

ASYNCEXCEPTION

此属性确定 WebSphere MQ JMS 类是仅在连接中断时或在 JMS API 调用异步发生任何异常时才通知 ExceptionListener。这适用于从此 ConnectionFactory 创建的所有已注册 ExceptionListener 的连接。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :ASYNCEXCEPTION

JMS 管理工具短名称 :AEX

以编程方式访问

Setters/Getters

- MQConnectionFactory。setAsync 异常 ()
- MQConnectionFactory。getAsyncExceptions ()

值

ASYNC_EXCEPTIONS_ALL

在同步 API 调用作用域外部异步检测到的任何异常以及所有连接中断异常都会发送到 ExceptionListener。

| 环境 | 值 |
|-----------------------|--|
| JMS 管理工具 | ALL |
| 程序化 | WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1 |
| WebSphere MQ Explorer | 全部 |

ASYNC_EXCEPTIONS_CONNECTIONBROKEN

只有指示连接中断的异常才会发送到 ExceptionListener。在异步处理期间发生的任何其他异常都不会报告给 ExceptionListener，因此，不会向应用程序通知这些异常。**V7.5.0.8** 这是 IBM WebSphere MQ Version 7.5.0 修订包 8 中的缺省值 (请参阅 [JMS: Version 7.5](#) 中的异常侦听器更改)。

| 环境 | 值 |
|-----------------------|--|
| JMS 管理工具 | 连接中断 |
| 程序化 | WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1 |
| WebSphere MQ Explorer | 连接中断 |

定义了以下附加常量：**V7.5.0.8**

- 从 Version 7.5.0, Fix Pack 8: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN
- 在 Version 7.5.0, Fix Pack 8 之前: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

相关概念

[WebSphere MQ classes for JMS 中的异常](#)

BROKERCCDURSUBQ

针对 ConnectionConsumer 从中检索持久预订消息的队列的名称。

适用对象

Topic

JMS 管理工具长名称 :BROKERCCDURSUBQ

JMS 管理工具短名称 :CCDSUB

以编程方式访问

Setter 方法/getter 方法

- MQTopic.setBrokerCCDurSubQueue()

- MQTopic.getBrokerCCDurSubQueue()

值

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE

这是缺省值。

任何有效字符串

BROKERCCSUBQ

从其中检索 ConnectionConsumer 的非持久预订消息的队列的名称。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :BROKERCCSUBQ

JMS 管理工具短名称 :CCSUB

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

值

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE

这是缺省值。

任何有效字符串

BROKERCONQ

代理的控制队列名称。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :BROKERCONQ

JMS 管理工具短名称 :BCON

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

值

SYSTEM.BROKER.CONTROL.QUEUE

这是缺省值。

任何有效字符串

BROKERDURSUBQ

在 WebSphere MQ 消息传递提供程序迁移方式中使用 WebSphere MQ JMS 类时，此属性指定从中检索持久预订消息的队列的名称。

适用对象

Topic

JMS 管理工具长名称 :BROKERDURSUBQ

JMS 管理工具短名称 :BDSUB

以编程方式访问

Setter 方法/getter 方法

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

值

SYSTEM.JMS.D.SUBSCRIBER.QUEUE

这是缺省值。

任何有效字符串

从 SYSTEM.JMS.D

相关概念

[选择 WebSphere MQ 消息传递提供程序方式的规则](#)

BROKERPUBQ

发送发布的消息的队列的名称（流队列）。

适用对象

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :BROKERPUBQ

JMS 管理工具短名称 :BPUB

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

值

SYSTEM.BROKER.DEFAULT.STREAM

这是缺省值。

任何有效字符串

BROKERPUBQMGR

队列管理器的名称，该队列管理器拥有其中发送主题上发布的消息的队列。

适用对象

Topic

JMS 管理工具长名称 :BROKERPUBQMGR

JMS 管理工具短名称 :BPQM

以编程方式访问

Setter 方法/getter 方法

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

值

null

这是缺省值。

任何有效字符串

BROKERQMGR

运行代理程序的队列管理器的名称。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :BROKERQMGR

JMS 管理工具短名称 :BQM

以编程方式访问

设置者/获取者

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

值

null

这是缺省值。

任何有效字符串

BROKERSUBQ

在 WebSphere MQ 消息传递提供程序迁移方式中使用 WebSphere MQ classes for JMS 时, 此属性指定从中检索非持久预订消息的队列的名称。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :BROKERSUBQ

JMS 管理工具短名称 :BSUB

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

值

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE

这是缺省值。

任何有效字符串

从 SYSTEM.JMS.ND

相关概念

[选择 WebSphere MQ 消息传递提供程序方式的规则](#)

BROKERVER

正在使用的代理程序的版本。

适用对象

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :BROKERVER

JMS 管理工具短名称 :BVER

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setBrokerVersion ()
- MQConnectionFactory.getBrokerVersion ()

值

V1

使用 WebSphere MQ 发布/预订代理，或者以兼容性方式使用 WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker 或 WebSphere Business Integration Message Broker 的代理。如果 TRANSPORT 设置为 BIND 或 CLIENT, 那么这是缺省值。

V2

以本机方式使用 WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker 或 WebSphere Business Integration Message Broker 的代理。如果 TRANSPORT 设置为 DIRECT 或 DIRECTHTTP, 那么这是缺省值。

未指定

在代理从 V6 迁移到 V7 之后, 请设置此属性, 以便不再使用 RFH2 头。迁移后, 此属性不再相关。

CCDTURL

统一资源定位符 (URL), 用于标识包含客户机通道定义表的文件的名称和位置, 以及指定可以如何访问此文件。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CCDTURL

JMS 管理工具短名称 :CCDT

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

值

null

这是缺省值。

统一资源定位符 (URL)

CCSID

要用于连接或目标的编码字符集标识。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CCSID

JMS 管理工具短名称 :CCS

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setCCSID()
- MQConnectionFactory.getCCSID()

值

819

这是连接工厂的缺省值。

1208

这是目标的缺省值。

任何正整数

通道

正在使用的客户机连接通道的名称。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CHANNEL

JMS 管理工具短名称: 陈

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

值

SYSTEM.DEF.SVRCONN

这是缺省值。

任何有效字符串

CLEANUP

BROKER 或 MIGRATE 预订存储的清除级别。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CLEANUP

JMS 管理工具短名称 :CL

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setCleanupLevel ()
- MQConnectionFactory。getCleanupLevel ()

值

安全

使用安全清除。这是缺省值。

ASPROP

根据 Java 命令行上设置的属性, 使用安全, 强清除或不清除。

无

不使用清除。

STRONG

使用强清除。

CLEANUPINT

两次后台执行发布/预订清除实用程序之间的时间间隔 (以毫秒计)。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CLEANUPINT

JMS 管理工具短名称 :CLINT

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setCleanup 时间间隔 ()

- MQConnectionFactory。getCleanupInterval ()

值

3600000

这是缺省值。

任何正整数

connectionNameList

TCP/IP 连接名称的列表。将按顺序尝试该列表，每次重新连接重试一次。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :CONNECTIONNAMELIST

JMS 管理工具短名称 :CNLIST

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setconnectionNameList ()
- MQConnectionFactory。getconnectionNameList ()

值

以逗号分隔的 HOSTNAME (PORT) 列表。HOSTNAME 可以是 DNS 名称或 IP 地址。

PORT 在缺省情况下为 1414。

CLIENTRECONNECTOPTIONS

用于管理重新连接的选项。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :CLIENTRECONNECTOPTIONS

JMS 管理工具短名称 :CROPT

以编程方式访问

设置者/获取者

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

值

QMGR

应用程序可以重新连接，但仅重新连接至最初连接的队列管理器。

如果可以重新连接应用程序，但 WebSphere MQ classes for JMS 应用程序与它首次建立连接的队列管理器之间存在亲缘关系，请使用此值。

如果希望应用程序自动重新连接到高可用性队列管理器的备用实例，请选择此值。

要以编程方式使用此值，请使用常量 `WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR`。

ANY

应用程序可重新连接至任何队列管理器。

仅当用于 JMS 应用程序的 WebSphere MQ 类与最初与其建立连接的队列管理器之间没有亲缘关系时，才使用重新连接选项。

要使用程序中的此值，请使用常量 `WMQConstants.WMQ_CLIENT_RECONNECT`。

DISABLED

将不重新连接应用程序。

要以编程方式使用此值，请使用常量 `WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED`。

ASDEF

应用程序是否将自动重新连接取决于 WebSphere MQ 通道属性 `DefReconnect` 的值。

要使用程序中的此值，请使用常量 `WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF`。

CLIENTRECONNECTTIMEOUT

停止重新连接重试之前的时间。

适用对象

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`

JMS 管理工具长名称 :`CLIENTRECONNECTTIMEOUT`

JMS 管理工具短名称 :`CRT`

以编程方式访问

Setter 方法/getter 方法

- `MQConnectionFactory.setClientReconnectTimeout()`
- `MQConnectionFactory.setClientReconnectTimeout()`

值

时间间隔 (以秒计)。缺省值 1800 (30 分钟)。

CLIENTID

客户机标识用来唯一地标识持久预订的应用程序连接。

适用对象

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`

JMS 管理工具长名称 :`CLIENTID`

JMS 管理工具短名称 :`CID`

以编程方式访问

Setter 方法/getter 方法

- `MQConnectionFactory.setClient 标识 ()`
- `MQConnectionFactory.getClientId ()`

值

null

这是缺省值。

任何有效字符串

CLONESUPP

同一持久主题订户的两个或更多实例是否可以同时运行。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CLONESUPP

JMS 管理工具短名称 :CLS

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setClone 支持 ()
- MQConnectionFactory.getClone 支持 ()

值

DISABLED

一次只能运行持久主题订户的一个实例。这是缺省值。

ENABLED

同一持久主题订户的两个或多个实例可以同时运行，但每个实例必须在单独的 Java 虚拟机 (JVM) 中运行。

COMPHDR

可用于压缩连接上的头数据的方法列表。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :COMPHDR

JMS 管理工具短名称 :HC

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

值

无

这是缺省值。

SYSTEM

执行 RLE 消息头压缩。

COMPMSG

可用于压缩连接上的消息数据的方法列表。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :COMPMSG

JMS 管理工具短名称 :MC

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

值

无

这是缺省值。

以空白字符分隔的下列一个或多个值的列表:

RLE ZLIBFAST ZLIBHIGH

CONNOPT

控制使用绑定传输的 JMS 应用程序的 WebSphere MQ 类如何连接到队列管理器。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CONNOPT

JMS 管理工具短名称 :CNOPT

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMQConnection 选项 ()
- MQConnectionFactory.getMQConnection 选项 ()

值

标准

应用程序与队列管理器之间绑定的性质取决于队列管理器的 *DefaultBindType* 属性的值, STANDARD 值映射到 WebSphere MQ *ConnectOption* MQCNO_STANDARD_BINDING

SHARED

应用程序和本地队列管理器代理程序在不同的执行单元中运行, 但共享一些资源。此值映射到 WebSphere MQ *ConnectOption* MQCNO_SHARED_BINDING。

隔离

应用程序和本地队列管理器代理程序在单独的执行单元中运行, 并且不共享任何资源。ISOLATION 值映射到 WebSphere MQ *ConnectOption* MQCNO_SOLATED_BINDING。

FASTPATH

应用程序和本地队列管理器代理程序在同一执行单元中运行。此值映射到 WebSphere MQ *ConnectOption* MQCNO_FASTPATH_BINDING。

SERIALQM

应用程序请求在队列管理器的作用域内独占使用连接标记。此值映射到 WebSphere MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR。

SERIALQSG

应用程序请求在队列管理器所属的队列共享组的作用域内独占使用连接标记。SERIALQSG 值映射到 WebSphere MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_QSG。

RESTRICTQM

应用程序请求共享使用连接标记，但在队列管理器的作用域内共享使用连接标记存在限制。此值映射到 WebSphere MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_Q_MGR。

RESTRICTQSG

应用程序请求共享使用连接标记，但在队列管理器所属的队列共享组的作用域内共享使用连接标记存在限制。此值映射到 WebSphere MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_QSG。

有关 WebSphere MQ 连接选项的更多信息，请参阅 [使用 MQCONNX 调用连接到队列管理器](#)

CONNTAG

当应用程序连接到队列管理器时，队列管理器与工作单元中应用程序更新的资源相关联的标记。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :CONNTAG

JMS 管理工具短名称 :CNTAG

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setConnTag ()
- MQConnectionFactory.getConnTag ()

值

由 **128** 个元素组成的字节数组，其中每个元素都是 **0**
这是缺省值。

任何字符串

如果该值的长度超过 128 个字节，那么将截断该值。

DESCRIPTION

所存储对象的描述。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :DESCRIPTION

JMS 管理工具短名称 :DESC

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

值

null

这是缺省值。

任何有效字符串

DIRECTAUTH

是否在与代理程序的实时连接上使用 SSL 认证。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :DIRECTAUTH

JMS 管理工具短名称 :DAUTH

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setDirectAuth ()
- MQConnectionFactory。getDirectAuth ()

值

基本

无认证, 用户名认证或密码认证。这是缺省值。

CERTIFICATE

公用密钥证书认证。

ENCODING

将消息发送到此目标时, 如何表示消息体中的数字数据。此属性指定二进制整数、压缩十进制整数和浮点数的表示法。

适用对象

队列, 主题

JMS 管理工具长名称 :ENCODING

JMS 管理工具短名称 :ENC

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setEncoding()
- MQDestination.getEncoding()

值

Encoding 属性

ENCODING 属性可以采用的有效值是从三个子属性构造的:

整型编码

正常或反转

十进制编码

正常或反转

浮点编码

IEEE 正常, IEEE 逆向或 z/OS

ENCODING 属性表示为具有以下语法的三字符字符串:

```
{N|R}{N|R}{N|R|3}
```

在此字符串中:

- N 表示正常
- R 表示反转
- 3 表示 z/OS
- 第一个字符表示 整数编码
- 第二个字符表示 十进制编码
- 第三个字符表示 浮点编码

这将为 ENCODING 属性提供包含 12 个可能值的集合。

还有一个附加值, 即字符串 NATIVE, 用于为 Java 平台设置相应的编码值。

以下示例显示了 ENCODING 的有效组合:

```
ENCODING(NNR)  
ENCODING(NATIVE)  
ENCODING(RR3)
```

EXPIRY

目标上的消息到期的时间。

适用对象

队列, 主题

JMS 管理工具长名称:到期

JMS 管理工具短名称 :EXP

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setExpiry()
- MQDestination.getExpiry()

值

APP

到期可以由 JMS 应用程序定义。这是缺省值。

UNLIM

不会发生到期。

0

不会发生到期。

表示到期的任何正整数 (以毫秒计)。

FAILIFQUIESCE

此属性确定如果队列管理器处于停顿状态, 或者应用程序正在使用 CLIENT 传输连接到队列管理器, 并且应用程序正在使用的通道已进入停顿状态 (例如, 使用 **STOP CHANNEL** 或 **STOP CHANNEL MODE(QUIESCE)** MQSC 命令), 那么对某些方法的调用是否失败。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称: FAILIFQUIESCE

JMS 管理工具短名称 :FIQ

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

值

YES

如果队列管理器处于停顿状态, 或者用于连接到队列管理器的通道处于停顿状态, 那么对某些方法的调用将失败。如果应用程序检测到其中任一情况, 那么应用程序可以完成其立即执行的任务并关闭连接, 从而允许队列管理器或通道实例停止。这是缺省值。

否

由于队列管理器或用于连接到队列管理器的通道处于停顿状态, 因此没有方法调用失败。如果指定此值, 那么应用程序无法检测到队列管理器或通道正在停顿。应用程序可能会继续对队列管理器执行操作, 因此会阻止队列管理器停止。

HOSTNAME

对于与队列管理器的连接, 是运行队列管理器的系统的主机名或 IP 地址, 或者对于与代理的实时连接, 是运行代理的系统的主机名或 IP 地址。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :HOSTNAME

JMS 管理工具短名称 :HOST

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setHost 名称 ()
- MQConnectionFactory.getHost 名称 ()

值

localhost

这是缺省值。

任何有效字符串

LOCALADDRESS

对于与队列管理器的连接，此属性指定要使用的本地网络接口或要使用的本地端口或本地端口范围。对于与代理的实时连接，此属性仅在使用多点广播时相关，并指定要使用的本地网络接口。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :LOCALADDRESS

JMS 管理工具短名称 :LA

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setLocalAddress ()
- MQConnectionFactory.getLocalAddress ()

值

"" (空字符串)

这是缺省值。

格式为 [ip-addr] [(低端口 [, 高端口])] 的字符串

以下是一些示例：

192.0.2.0

通道绑定到本地地址 192.0.2.0。

192.0.2.0(1000)

通道在本地绑定到地址 192.0.2.0，并使用端口 1000。

192.0.2.0(1000,2000)

通道在本地绑定到地址 192.0.2.0，并使用 1000 到 2000 范围内的端口。

(1000)

通道绑定到本地端口 1000。

(1000,2000)

通道在本地绑定到 1000 到 2000 范围内的端口。

您可以指定主机名而不是 IP 地址。对于与代理程序的实时连接，仅当使用多点广播时，此属性才相关，并且此属性的值不得包含端口号或端口号范围。在这种情况下，属性的有效值只有 null，IP 地址或主机名。

MAPNAMESTYLE

允许将兼容性样式用于 MapMessage 元素名称。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :MAPNAMESTYLE

JMS 管理工具短名称 :MNST

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

值

标准

将使用标准 com.ibm.jms.JMSMapMessage 元素命名格式。这是缺省值，允许将非合法 Java 标识用作元素名称。

兼容

将使用较旧的 com.ibm.jms.JMSMapMessage 元素命名格式。只有合法的 Java 标识可以用作元素名称。仅当将映射消息发送到使用版本低于 V 5.3 的 IBM WebSphere MQ classes for JMS 版本的应用程序时，才需要执行此操作。

MAXBUFFSIZE

等待应用程序处理时可以存储在内部消息缓冲区中的最大接收消息数。仅当 TRANSPORT 的值为 DIRECT 或 DIRECTHTTP 时，此属性才适用。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :MAXBUFFSIZE

JMS 管理工具短名称 :MBSZ

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

值

1000

这是缺省值。

任何正整数

MDREAD

此属性确定 JMS 应用程序是否可以抽取 MQMD 字段的值。

适用对象

JMS 管理工具长名称 :MDREAD

JMS 管理工具短名称 :MDR

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

值

否

发送消息时，将不会更新所发送消息中的 JMS_IBM_MQMD* 属性，从而不会反映 MQMD 中已更新的字段值。接收消息时，在所接收到的消息中不存在任何 JMS_IBM_MQMD* 属性，即使发送方已设置了这些属性的一部分或全部也是如此。这是管理工具的缺省值。

对于程序，请使用 False。

Yes

发送消息时，将更新已发送消息上的所有 JMS_IBM_MQMD* 属性，以反映 MQMD 中的已更新字段值，包括发送方未显式设置的属性。接收消息时，收到的消息上提供了所有 JMS_IBM_MQMD* 属性，包括发送方未显式设置的属性。

对于程序，请使用 True。

MDWRITE

此属性确定 JMS 应用程序是否可以设置 MQMD 字段的值。

适用对象

队列，主题

JMS 管理工具长名称 :MDWRITE

JMS 管理工具短名称 :MDR

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

值

否

将忽略所有 JMS_IBM_MQMD* 属性，并且不会将它们值复制到底层的 MQMD 结构。这是管理工具的缺省值。

对于程序，请使用 False。

YES

将处理 JMS_IBM_MQMD* 属性。它们的值将复制到底层的 MQMD 结构。

对于程序，请使用 True。

MDMSGCTX

JMS 应用程序要设置的消息上下文级别。应用程序必须以相应的上下文权限运行才能使属性生效。

适用对象

JMS 管理工具长名称 :MDMSGCTX

JMS 管理工具短名称 :MDCTX

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

值

Default

MQOPEN API 调用和 MQPMO 结构未指定显式消息上下文选项。这是管理工具的缺省值。

对于程序，请使用 WMQ_MDCTX_DEFAULT。

SET_IDENTITY_CONTEXT

MQOPEN API 调用指定消息上下文选项 MQOO_SET_IDENTITY_CONTEXT，而 MQPMO 结构指定 MQPMO_SET_IDENTITY_CONTEXT。

对于程序，请使用 WMQ_MDCTX_SET_IDENTITY_CONTEXT。

SET_ALL_CONTEXT

MQOPEN API 调用指定消息上下文选项 MQOO_SET_ALL_CONTEXT，而 MQPMO 结构指定 MQPMO_SET_ALL_CONTEXT。

对于程序，请使用 WMQ_MDCTX_SET_ALL_CONTEXT。

MSGBATCHSZ

使用异步消息传递时要从一个包中的队列获取的最大消息数。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :MAXBUFFSIZE

JMS 管理工具短名称 :MBSZ

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

值

10

这是缺省值。

任何正整数

MSGBODY

确定 JMS 应用程序是否作为消息有效内容的一部分访问 IBM WebSphere MQ 消息的 MQRFH2。

适用对象

队列, 主题

JMS 管理工具长名称 :WMQ_MESSAGE_BODY

JMS 管理工具短名称 :MBODY

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

值

未指定

发送时，IBM WebSphere MQ classes for JMS 是否生成并包含 MQRFH2 头取决于 WMQ_TARGET_CLIENT 的值。接收时，充当值 JMS。

JMS

发送时，IBM WebSphere MQ classes for JMS 会自动生成 MQRFH2 头，并将其包含在 WebSphere MQ 消息中。

接收时，IBM WebSphere MQ classes for JMS 会根据 MQRFH2 中的值设置 JMS 消息属性 (如果存在); 它不会将 MQRFH2 作为 JMS 消息体的一部分提供。

MQ

发送时，IBM WebSphere MQ classes for JMS 不生成 MQRFH2。

接收时，IBM WebSphere MQ classes for JMS 会将 MQRFH2 显示为 JMS 消息体的一部分。

MSGRETENTION

连接使用者是否将未传递的消息保留在输入队列中。

适用对象

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,
JMS 管理工具长名称 :MSGRETENTION
JMS 管理工具短名称 :MRET

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setMessage 保留时间 ()
- MQConnectionFactory。getMessageRetention ()

值

Yes

未传递的消息仍保留在输入队列中。这是缺省值。

否

未传递的消息将根据其处置选项进行处理。

MSGSELECTION

确定消息选择是由 WebSphere MQ classes for JMS 还是由代理程序完成。如果 TRANSPORT 的值为 DIRECT, 那么消息选择始终由代理完成, 并且将忽略 MSGSELECTION 的值。当 BROKERVER 具有值 V1 时, 不支持代理选择消息。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory
JMS 管理工具长名称 :MSGSELECTION

JMS 管理工具短名称 :MSEL

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMessageSelection ()
- MQConnectionFactory.getMessageSelection ()

值

CLIENT

消息选择由 WebSphere MQ classes for JMS 完成。这是缺省值。

BROKER

消息选择由代理程序完成。

MULTICAST

在与代理程序的实时连接上启用多点广播，并指定使用多点广播将消息从代理程序传递到消息使用者的精确方式(如果已启用)。此属性不会影响消息生产者将消息发送到代理的方式。

适用对象

ConnectionFactory, TopicConnectionFactory, 主题

JMS 管理工具长名称:多点广播

JMS 管理工具短名称 :MCAST

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

值

DISABLED

不使用多点广播传输将消息传递给消息使用者。这是 ConnectionFactory 和 TopicConnectionFactory 对象的缺省值。

ASCF

根据与消息使用者相关联的连接工厂的多点广播设置，将消息传递到消息使用者。在创建消息使用者时，将记录连接工厂的多点广播设置。此值仅对 Topic 对象有效，并且是 Topic 对象的缺省值。

ENABLED

如果在代理中为多点广播配置了主题，那么将使用多点广播传输将消息传递到消息使用者。如果对主题配置了可靠的多点广播，那么将使用可靠的服务质量。

可靠

如果为代理中的可靠多点广播配置了主题，那么将使用具有可靠服务质量的多点广播传输将消息传递给消息使用者。如果没有对主题配置可靠的多点广播，那么无法为该主题创建消息使用者。

无

如果在代理中为多点广播配置了主题，那么将使用多点广播传输将消息传递到消息使用者。即使对主题配置了可靠的多点广播，也不会使用可靠服务质量。

OPTIMISTIC PUBLICATION

此属性确定 WebSphere MQ classes for JMS 是将控制权立即返回给已发布消息的发布程序，还是仅在完成与调用关联的所有处理并将结果报告给发布程序后才返回控制权。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :OPTIMISTICPUBLICATION

JMS 管理工具短名称 :OPTPUB

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setOptimisticPublication ()
- MQConnectionFactory。getOptimisticPublication ()

值

否

当发布者发布消息时, WebSphere MQ classes for JMS 不会将控制权返回给发布者, 直到它完成了与调用相关联的所有处理并可以向发布者报告结果。这是缺省值。

YES

当发布者发布消息时, WebSphere MQ classes for JMS 在完成与调用关联的所有处理并向发布者报告结果之前, 会立即向发布者返回控制权。WebSphere MQ classes for JMS 仅在发布程序落实消息时报告结果。

OUTCOMENOTIFICATION

此属性确定 WebSphere MQ classes for JMS 是立即向刚刚确认或落实消息的订户返回控制, 还是仅在完成与调用关联的所有处理并向订户报告结果后返回控制。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称:OUTCOMENO 分层

JMS 管理工具短名称 :NOTIFY

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setOutcomeNotification ()
- MQConnectionFactory。getOutcomeNotification ()

值

YES

当订户确认或落实消息时, WebSphere MQ classes for JMS 不会将控制权返回给订户, 直到它完成与调用关联的所有处理并可以向订户报告结果为止。这是缺省值。

否

当订户确认或落实消息时, WebSphere MQ classes for JMS 会在完成与调用关联的所有处理并向订户报告结果之前, 立即向订户返回控制权。

PERSISTENCE

发送到目标的消息的持久性。

适用对象

队列, 主题

JMS 管理工具长名称 :PERSISTENCE

JMS 管理工具短名称 :PER

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setPersistence()
- MQDestination.getPersistence()

值

APP

持久性由 JMS 应用程序定义。这是缺省值。

qdef

持久性采用队列缺省值。

pers

消息是持久消息。

NON

消息是非持久消息。

高

请参阅 [JMS 持久消息](#) , 以获取有关使用此值的更多信息。

POLLINGINT

如果会话中的每个消息侦听器在其队列中没有合适的消息, 那么这是在每个消息侦听器再次尝试从其队列中获取消息之前经过的最大时间间隔 (以毫秒为单位)。如果没有合适的消息可用于会话中的任何消息侦听器的情况频繁发生, 请考虑增大此属性的值。仅当 TRANSPORT 的值为 BIND 或 CLIENT 时, 此属性才相关。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :POLLINGINT

JMS 管理工具短名称 :PINT

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setPolling 时间间隔 ()
- MQConnectionFactory。getPollingInterval ()

值

5000

这是缺省值。

任何正整数

端口

对于与队列管理器的连接，这是队列管理器正在侦听的端口号，对于与代理程序的实时连接，这是代理程序正在侦听实时连接的端口号。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :PORT

JMS 管理工具短名称 :PORT

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setPort()
- MQConnectionFactory.getPort()

值

1414

如果 TRANSPORT 设置为 CLIENT，那么这是缺省值。

1506

如果 TRANSPORT 设置为 DIRECT 或 DIRECTHTTP，那么这是缺省值。

任何正整数

PRIORITY

发送到目标的消息的优先级。

适用对象

队列, 主题

JMS 管理工具长名称 :PRIORITY

JMS 管理工具短名称 :PRI

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setPriority()
- MQDestination.getPriority()

值

APP

优先级由 JMS 应用程序定义。这是缺省值。

qdef

优先级采用队列缺省值。

范围 0-9 中的任何整数

从最低到最高。

PROCESSDURATION

此属性确定订户是否保证在将控制权返回给 WebSphere MQ classes for JMS 之前快速处理其接收到的任何消息。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :PROCESSDURATION

JMS 管理工具短名称 :PROCDUR

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setProcessDuration ()
- MQConnectionFactory。getProcessDuration ()

值

未知

订户无法保证它能够以多快的速度处理它接收到的任何消息。这是缺省值。

SHORT

订户保证在将控制权返回给 WebSphere MQ classes for JMS 之前快速处理其接收到的任何消息。

PROVIDERVERSION

此属性区分两种 WebSphere MQ 消息传递操作方式: WebSphere MQ 消息传递提供程序正常方式和 WebSphere MQ 消息传递提供程序迁移方式。

WebSphere MQ 消息传递提供程序正常方式使用 WebSphere MQ V 7.0 队列管理器的所有功能来实现 JMS。此方式仅用于连接到 WebSphere MQ 队列管理器, 并且可以在客户机或绑定方式下连接到 WebSphere MQ Version 7.0 队列管理器。此方式优化为使用新的 WebSphere MQ V 7.0 功能。如果不使用 WebSphere MQ 实时传输, 那么使用的操作方式主要由连接工厂的 PROVIDERVERSION 属性确定。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :PROVIDERVERSION

JMS 管理工具短名称 :PVER

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setProvider 版本 ()
- MQConnectionFactory.getProvider 版本 ()

值

可以将 **PROVIDERVERSION** 设置为可能的值: 7, 6 或 *unspecified*。但是, **PROVIDERVERSION** 可以是以下任何一种格式的字符串:

- V.R.M.F
- V.R.M
- V.R

- V

其中 V、R、M 和 F 是大于或等于零的整数值。

7

使用 WebSphere MQ 消息传递提供程序正常方式。

如果将 PROVIDERVERSION 设置为 7，那么只有 WebSphere MQ 消息传递提供程序正常操作方式可用。如果由于连接工厂中的其他设置而连接到的队列管理器不是 V 7.0 队列管理器，那么 createConnection() 方法将因异常而失败。

WebSphere MQ 消息传递提供程序正常方式使用共享对话功能，可共享的对话数由服务器连接通道上的 SHARECNV () 属性控制。如果此属性设置为 0，那么您无法使用 WebSphere MQ 消息传递提供程序正常方式，并且 createConnection() 方法因异常而失败。

6

使用 WebSphere MQ 消息传递提供程序迁移方式。

WebSphere MQ JMS 类使用 WebSphere MQ Version 6.0 随附的功能和算法。如果要使用 WebSphere MQ Enterprise Transport 连接到 WebSphere Event Broker 或 WebSphere Message Broker，那么必须使用此方式。可以使用此方式连接到 WebSphere MQ V 7.0 队列管理器，但不会使用 V 7.0 队列管理器的任何新功能，例如，预读或流式方法。

未指定

这是缺省值，实际文本为 "unspecified"。

当连接工厂与新版本的 WebSphere MQ JMS 类配合使用时，使用 JNDI 中先前版本的 WebSphere MQ JMS 类创建的连接工厂将采用此值。以下算法用于确定所使用的操作方式。当调用 createConnection() 方法并使用连接工厂的其他方面来确定是否需要 WebSphere MQ 消息传递提供程序正常方式或 WebSphere MQ 消息传递提供程序迁移方式时，将使用此算法。

- 首先，尝试使用 WebSphere MQ 消息传递提供程序正常方式。
- 如果已连接的队列管理器不是 WebSphere MQ V 7.0，那么将关闭连接并改为使用 WebSphere MQ 消息传递提供程序迁移方式。
- 如果服务器连接通道上的 SHARECNV () 属性设置为 0，那么将关闭连接，并改为使用 WebSphere MQ 消息传递提供程序迁移方式。
- 如果 BROKERVER 设置为 1 或新的缺省 "unspecified" 值，那么将继续使用 WebSphere MQ 消息传递提供程序正常方式，因此任何发布/预订操作都将使用新的 WebSphere MQ V7.0 功能部件。如果以兼容性方式使用 WebSphere Event Broker 或 WebSphere Message Broker (并且您希望使用 V 6.0 发布/预订功能而不是 WebSphere MQ V 7 发布/预订功能)，请将 PROVIDERVERSION 设置为 6 确保使用 WebSphere MQ 消息传递提供程序迁移方式。

PROXYHOSTNAME

使用通过代理服务器与代理的实时连接时，代理服务器正在其上运行的系统的主机名或 IP 地址。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :PROXYHOSTNAME

JMS 管理工具短名称 :PHOST

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setProxyHostName()
- MQConnectionFactory.getProxyHostName()

值

null

代理服务器的主机名。这是缺省值。

PROXYPORT

使用通过代理服务器与代理的实时连接时，代理服务器正在侦听的端口号。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :PROXYPORT

JMS 管理工具短名称 :PPORT

以编程方式访问

Setter 方法/getter 方法

MQConnectionFactory.setProxy 端口 ()

MQConnectionFactory.getProxy 端口 ()

值

443

代理服务器的端口号。这是缺省值。

PUBACKINT

在 WebSphere MQ JMS 类请求来自代理的应答之前，发布者发布的消息数。

当您降低此属性的值时，WebSphere MQ classes for JMS 会更频繁地请求应答，因此发布程序的性能会下降。当增大该值时，WebSphere MQ classes for JMS 将花费更长时间在代理程序失败时抛出异常。仅当 TRANSPORT 的值为 BIND 或 CLIENT 时，此属性才相关。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :PROXYPORT

JMS 管理工具短名称 :PPORT

以编程方式访问

Setter 方法/getter 方法

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

值

25

任何正整数都可以是缺省值。

PUTASYNCALOWED

此属性确定是否允许消息生产者使用异步放置来将消息发送到此目标。

适用对象

队列, 主题

JMS 管理工具长名称 :PUTASYNCALLOWED

JMS 管理工具短名称 :PAALD

以编程方式访问

Setter 方法/getter 方法

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

值

AS_DEST

通过参考队列或主题定义来确定是否允许异步放置。这是缺省值。

AS_Q_DEF

通过参考队列定义来确定是否允许异步放置。

AS_TOPIC_DEF

通过参考主题定义来确定是否允许异步放置。

否

不允许异步放置。

YES

允许异步放置。

QMANAGER

要连接到的队列管理器的名称。

但是, 如果应用程序使用客户机通道定义表来连接到队列管理器, 请参阅 [将客户机通道定义表与 WebSphere MQ classes for JMS 配合使用](#)。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :QMANAGER

JMS 管理工具短名称 :QMGR

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setQueueManager ()
- MQConnectionFactory.getQueueManager ()

值

"" (空字符串)

任何字符串都可以是缺省值。

队列

JMS 队列目标的名称。这与队列管理器所使用的队列的名称相匹配。

适用对象

队列

JMS 管理工具长名称 :QUEUE

JMS 管理工具短名称 :QU

值

任何字符串

任何有效的 IBM WebSphere MQ 队列名称。

相关概念

[用于命名 IBM WebSphere MQ 对象的规则](#)

READAHEADALLOWED

此属性确定是否允许消息使用者和队列浏览器在接收非持久消息之前使用预读从该目标到内部缓冲区。

适用对象

队列, 主题

JMS 管理工具长名称 :READAHEADALLOWED

JMS 管理工具短名称 :RAALD

以编程方式访问

Setter 方法/getter 方法

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

值

AS_DEST

通过参考队列或主题定义来确定是否允许预读。这是管理工具中的缺省值。

在程序中使用 WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST。

AS_Q_DEF

通过参考队列定义来确定是否允许预读。

在程序中使用 WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF。

AS_TOPIC_DEF

通过参考主题定义来确定是否允许预读。

在程序中使用 WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF。

否

不允许预读。

在程序中使用 WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED。

YES

允许预读。

在程序中使用 WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED。

READAHEADCLOSEPOLICY

对于传递到异步消息侦听器的消息, 当消息使用者关闭时, 内部预读缓冲区中的消息会发生什么情况。

适用对象

队列, 主题

JMS 管理工具长名称 :READAHEADCLOSEPOLICY

JMS 管理工具短名称 :RACP

以编程方式访问

Setter 方法/getter 方法

- `MQDestination.setReadAheadClosePolicy()`
- `MQDestination.getReadAheadClosePolicy()`

值

全部增量

在返回之前, 内部预读缓冲区中的所有消息都将传递到应用程序的消息侦听器。这是管理工具中的缺省值。

在程序中使用 `WMQConstants.WMQ_READ_AHEAD_DELIVERALL`。

当前增量

只有当前消息侦听器调用会在返回之前完成, 内部预读缓冲区中可能会留有一些消息, 随后将丢弃这些消息。

在程序中使用 `WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT`。

RECEIVECCSID

用于设置队列管理器消息转换的目标 CCSID 的目标属性。除非 `RECEIVECONVERSION` 设置为 `WMQ_RECEIVE_CONVERSION_QMGR`, 否则将忽略该值

适用对象

队列, 主题

JMS 管理工具长名称 :RECEIVECCSID

JMS 管理工具短名称 :RCCS

以编程方式访问

设置者/获取者

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

值

`WMQConstants.WMQ_RECEIVE_CC_SID_JVM_DEFAULT`

0 - 使用 `JVM Charset.defaultCharset`

1208

UTF-8

ccsid

受支持的编码字符集标识。

RECEIVECONVERSION

用于确定队列管理器是否将执行数据转换的目标属性。

适用对象

队列, 主题

JMS 管理工具长名称 :RECEIVECONVERSION

JMS 管理工具短名称 :RCNV

以编程方式访问

设置者/获取者

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

值

WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG

1 -仅在 JMS 客户机上执行数据转换。从 V7.0 开始的缺省值, 以及从 7.0.1.5 开始的缺省值 (包括此值)。

WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR

2 -在将消息发送到客户机之前, 在队列管理器上执行数据转换。从 V7.0 到 V7.0.1.4 (含) 的缺省值 (仅此值), 除非应用了 APAR IC72897。

RECEIVEISOLATION

此属性确定订户是否可以接收尚未在订户队列上落实的消息。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称 :RECEIVEISOLATION

JMS 管理工具短名称 :RCVISOL

值

已落实

订户仅接收订户队列上已落实的消息。这是管理工具中的缺省值。

在程序中使用 `WMQConstants.WMQ_RCVISOL_COMMITTED`。

未落实

订户可以接收尚未在订户队列上落实的消息。

在程序中使用 `WMQConstants.WMQ_RCVISOL_UNCOMMITTED`。

RECEXIT

标识要连续运行的通道接收出口或一系列接收出口。

可能需要其他配置才能使 IBM WebSphere MQ classes for JMS 找到接收出口。有关更多信息, 请参阅 [配置 IBM WebSphere MQ classes for JMS 以使用通道出口](#)。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :RECEXIT

JMS 管理工具短名称 :RCX

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setReceiveExit ()
- MQConnectionFactory。getReceiveExit ()

值

null

由一个或多个以逗号分隔的项组成的字符串，其中每个项都是：

- 实现 WMQReceiveExit 接口的类的名称 (对于以 Java 编写的通道接收出口)。
 - 格式为 *libraryName(entryPointName)* 的字符串 (对于未以 Java 编写的通道接收出口)。
- 这是缺省值。

RECEXITINIT

调用通道接收出口时传递到这些出口的用户数据。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :RECEXITINIT

JMS 管理工具短名称 :RCXI

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

值

null

由一个或多个用逗号分隔的用户数据项组成的字符串。这是缺省值。

REPLYTOSTYLE

确定如何构造接收到的消息中的 JMSReplyTo 字段。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :REPLYTOSTYLE

JMS 管理工具短名称 :RTOST

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

值

Default

等同于 MQMD。

RFH2

使用 RFH2 头中提供的值。如果在发送应用程序中设置了 JMSReplyTo 值，请使用该值。

MQMD

使用 MQMD 提供的值。此行为等同于 WebSphere MQ V 6.0.2.4 和 6.0.2.5 的缺省行为。

如果发送应用程序设置的 JMSReplyTo 值不包含队列管理器名称，那么接收队列管理器将在 MQMD 中插入其自己的名称。如果将此参数设置为 MQMD，那么您使用的应答队列位于接收队列管理器上。如果将此参数设置为 RFH2，那么您使用的应答队列位于发送应用程序最初设置的已发送消息的 RFH2 中指定的队列管理器上。

如果发送应用程序设置的 JMSReplyTo 值包含队列管理器名称，那么此参数的值不重要，因为 MQMD 和 RFH2 都包含相同的值。

RESCANINT

当点到点域中的消息使用者使用消息选择器来选择要接收的消息时，WebSphere MQ JMS 类将按照队列的 MsgDeliverySequence 属性所确定的顺序搜索 WebSphere MQ 队列以查找合适的消息。

在 WebSphere MQ classes for JMS 找到合适的消息并将其传递给使用者之后，WebSphere MQ classes for JMS 将从其在队列中的当前位置继续搜索下一条合适的消息。WebSphere MQ classes for JMS 将继续以这种方式搜索队列，直到它到达队列末尾，或者直到此属性的值所确定的时间间隔 (以毫秒为单位) 已到期为止。在每种情况下，WebSphere MQ JMS 类返回到队列的开头以继续搜索，新的时间间隔开始。

适用对象

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

JMS 管理工具长名称:RESCANINT

JMS 管理工具短名称:RINT

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setRescan 时间间隔 ()
- MQConnectionFactory。getRescan 时间间隔 ()

值

5000

任何正整数都可以是缺省值。

SECEXIT

标识通道安全出口。

可能需要其他配置才能使 IBM WebSphere MQ classes for JMS 找到安全出口。有关更多信息，请参阅 [配置 IBM WebSphere MQ classes for JMS 以使用通道出口](#)。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称:SECEXIT

JMS 管理工具短名称:SXC

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSecurityExit ()
- MQConnectionFactory。getSecurityExit ()

值

null

实现 WMQSecurityExit 接口的类的名称 (对于以 Java 编写的通道安全出口)。

格式为 *libraryName(entryPointName)* 的字符串 (对于未以 Java 编写的通道安全出口)。

SECEXITINIT

调用通道安全出口时传递到此出口的用户数据。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :SECEXITINIT

JMS 管理工具短名称: SCXI

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

值

null

任何字符串都可以是缺省值。

SENDCHECKCOUNT

在单个非事务性 JMS 会话中, 在检查异步放置错误之间允许的发送调用数。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :SENDCHECKCOUNT

JMS 管理工具短名称: SCC

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

值

null

任何字符串都可以是缺省值。

SENDEXIT

标识要连续运行的通道发送出口或一系列发送出口。

可能需要其他配置才能使 IBM WebSphere MQ classes for JMS 找到发送出口。有关更多信息，请参阅 [配置 IBM WebSphere MQ classes for JMS 以使用通道出口](#)。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称:SENDEXIT

JMS 管理工具短名称:SDX

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setSendExit ()
- MQConnectionFactory。getSendExit ()

值

null

由一个或多个以逗号分隔的项组成的任何字符串，其中每个项都是：

- 实现 WMQSendExit 接口的类的名称 (对于以 Java 编写的通道发送出口)。
- 格式为 *libraryName(entryPointName)* 的字符串 (对于未以 Java 编写的通道发送出口)。
-

这是缺省值。

SENDEXITINIT

调用通道发送出口时传递到此出口的用户数据。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称:SENDEXITINIT

JMS 管理工具短名称:SDXI

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSendExitInit()
- MQConnectionFactory.getSendExitInit()

值

null

包含一个或多个用逗号分隔的用户数据项的任何字符串都可以是缺省值。

SHARECONVALLOWED

此属性确定如果通道定义匹配，客户机连接是否可以与从同一进程到同一队列管理器的其他顶级 JMS 连接共享其套接字。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称: SHARECONVALLOWED

JMS 管理工具短名称: SCALD

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

值

YES

这是管理工具的缺省值。

对于程序，请使用 WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES。

否

此值用于管理工具。

对于程序，请使用 WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO。

SPARSESUBS

控制 TopicSubscriber 对象的消息检索策略。

适用对象

ConnectionFactory, TopicConnectionFactory

JMS 管理工具长名称: SPARSESUBS

JMS 管理工具短名称: SSUBS

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSparseSubscriptions ()
- MQConnectionFactory.getSparseSubscriptions ()

值

否

预订接收频繁的匹配消息。这是管理工具的缺省值。

对于程序，请使用 false。

YES

预订接收不常见的匹配消息。该值要求可以打开预订队列以供浏览。
对于程序，请使用 true。

SSLCIPHERSUITE

要用于 SSL 连接的密码套件。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :SSLCIPHERSUITE

JMS 管理工具短名称: SCPHS

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSSLCipher 套件 ()
- MQConnectionFactory.getSSLCipherSuite ()

值

null

这是缺省值。有关更多信息，请参阅 [JMS 对象的 SSL 属性](#)。

SSLCRL

用于检查 SSL 证书撤销的 CRL 服务器。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称: SSLCRL

JMS 管理工具短名称 :SCRL

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSSLCertStores ()
- MQConnectionFactory.getSSLCertStores ()

值

null

以空格分隔的 LDAP URL 列表。这是缺省值。有关更多信息，请参阅 [JMS 对象的 SSL 属性](#)。

SSLFIPSREQUIRED

此属性确定 SSL 连接是否必须使用 IBM Java JSSE FIPS 提供程序 (IBMJSSEFIPS) 支持的 CipherSuite 。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称: SSLFIPSREQUIRED

JMS 管理工具短名称: SFIPS

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setSSLFips 必需 ()
- MQConnectionFactory。getSSLFips 必需 ()

值

否

SSL 连接可以使用 IBM Java JSSE FIPS 提供程序 (IBMJSSEFIPS) 不支持的任何 CipherSuite。这是缺省值。在程序中, 使用 false。

YES

SSL 连接必须使用受 IBMJSSEFIPS 支持的 CipherSuite。在程序中, 使用 true。

SSLPEERNAME

对于 SSL, 必须与队列管理器提供的 专有名称 框架相匹配。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称: SSLPEERNAME

JMS 管理工具短名称: SPEER

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSSLPeer 名称 ()
- MQConnectionFactory。getSSLPeer 名称 ()

值

null

这是缺省值。有关更多信息, 请参阅 [JMS 对象的 SSL 属性](#)。

SSLRESETCOUNT

对于 SSL, 在重新协商用于加密的密钥之前, 连接发送和接收的字节总数。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称: SSLRESETCOUNT

JMS 管理工具短名称 :SRC

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory。setSSLReset 计数 ()
- MQConnectionFactory。getSSLReset 计数 ()

值

0

零或任何小于或等于 999，999 或 999 的正整数。这是缺省值。有关更多信息，请参阅 [JMS 对象的 SSL 属性](#)。

STATREFRESHINT

长时间运行的事务的刷新之间的时间间隔（以毫秒为单位），该事务检测订户何时断开与队列管理器的连接。

仅当 SUBSTORE 的值为 QUEUE 时，此属性才相关。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :STATREFRESHINT

JMS 管理工具短名称: SRI

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

值

6000

任何正整数都可以是缺省值。有关更多信息，请参阅 [JMS 对象的 SSL 属性](#)。

SUBSTORE

其中，WebSphere MQ JMS 类存储与活动预订相关的持久数据。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称: SUBSTORE

JMS 管理工具短名称 :SS

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setSubscriptionStore ()
- MQConnectionFactory。getSubscriptionStore ()

值

BROKER

使用基于代理的预订存储来保存预订的详细信息。这是管理工具的缺省值。

对于程序，请使用 `WMQConstants.WMQ_SUBSTORE_BROKER`。

MIGRATE

将预订信息从基于队列的预订存储传输到基于代理的预订存储。

对于程序，请使用 `WMQConstants.WMQ_SUBSTORE_MIGRATE`。

队列

使用基于队列的预订存储来保存预订的详细信息。

对于程序，请使用 `WMQConstants.WMQ_SUBSTORE_QUEUE`。

SYNCPOINTALLGETS

此属性确定是否要在同步点下执行所有获取。

适用对象

`ConnectionFactory`，`QueueConnectionFactory`，`TopicConnectionFactory`，`XAConnectionFactory`，`XAQueueConnectionFactory`，`XATopicConnectionFactory`

JMS 管理工具长名称: `SYNCPOINT-tallgets`

JMS 管理工具短名称: `SPAG`

以编程方式访问

Setter 方法/getter 方法

- `MQConnectionFactory.setSyncpointAllGets()`
- `MQConnectionFactory.getSyncpointAllGets()`

值

否

这是缺省值。

Yes

TARGCLIENT

此属性确定是否使用 WebSphere MQ RFH2 格式与目标应用程序交换信息。

适用对象

队列，主题

JMS 管理工具长名称: `TARGCLIENT`

JMS 管理工具短名称: `TC`

以编程方式访问

Setter 方法/getter 方法

- `MQDestination.setTargetClient()`
- `MQDestination.getTargetClient()`

值

JMS

消息的目标是 JMS 应用程序。这是管理工具的缺省值。

对于程序，请使用 `WMQConstants.WMQ_CLIENT_JMS_COMPLIANT`。

MQ

消息的目标是非 JMS WebSphere MQ 应用程序。

对于程序，请使用 `WMQConstants.WMQ_CLIENT_NONJMS_MQ`。

TARGCLIENTMATCHING

此属性确定发送到入局消息的 `JMSReplyTo` 头字段所标识的队列的应答消息是否只有在入局消息具有 `MQRFH2` 头时才具有 `MQRFH2` 头。

适用对象

`ConnectionFactory`, `QueueConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`

JMS 管理工具长名称: `TARGCLIENTMATCHING`

JMS 管理工具短名称: `TCM`

以编程方式访问

Setter 方法/getter 方法

- `MQConnectionFactory.setTargetClientMatching()`
- `MQConnectionFactory.getTargetClientMatching()`

值

YES

如果入局消息没有 `MQRFH2` 头，那么从消息的 `JMSReplyTo` 头字段派生的 `Queue` 对象的 `TARGCLIENT` 属性将发送到 MQ。如果消息具有 `MQRFH2` 头，那么 `TARGCLIENT` 属性将改为设置为 JMS。这是管理工具的缺省值。

对于程序，请使用 `true`。

否

从入局消息的 `JMSReplyTo` 头字段派生的 `Queue` 对象的 `TARGCLIENT` 属性始终设置为 JMS。

对于程序，请使用 `false`。

TEMPMODEL

从中创建 JMS 临时队列的模型队列的名称。

适用对象

`ConnectionFactory`, `QueueConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`

JMS 管理工具长名称: `TEMPMODEL`

JMS 管理工具短名称: `TM`

以编程方式访问

Setter 方法/getter 方法

- `MQConnectionFactory.setTemporary 模型 ()`
- `MQConnectionFactory.getTemporary 模型 ()`

值

SYSTEM.DEFAULT.MODEL.QUEUE

任何字符串都可以是缺省值。

TEMPQPREFIX

用于构成 WebSphere MQ 动态队列的名称的前缀。

适用对象

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

JMS 管理工具长名称:TEMPQPREFIX

JMS 管理工具短名称: TQP

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setTempQPrefix ()
- MQConnectionFactory。getTempQPrefix ()

值

"" (空字符串)

在 z/OS 上使用的前缀为 CSQ.*，在所有其他平台上使用的前缀为 AMQ.*。这些是缺省值。

队列前缀

队列前缀是符合在 WebSphere MQ 对象描述符 (结构 MQOD) 中构成 *DynamicQName* 字段内容的规则的任何字符串，但最后一个非空白字符必须是星号。

TEMPTOPICPREFIX

创建临时主题时，JMS 会生成格式为 "TEMP/TEMPTOPICPREFIX/*unique_id*" 的主题字符串，或者如果此属性保留缺省值，那么仅保留 "TEMP/*unique_id*"。指定非空 TEMPTOPICPREFIX 允许定义特定模型队列，用于为此连接下创建的临时主题的订户创建受管队列。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称:TEMPTOPICPREFIX

JMS 管理工具短名称: TTP

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

值

仅由 WebSphere MQ 主题字符串的有效字符组成的任何非空字符串。缺省值为 "" (空字符串)。

TOPIC

JMS 主题目标的名称，此值由队列管理器用作发布或预订的主题字符串。

适用对象

Topic

JMS 管理工具长名称: TOPIC

JMS 管理工具短名称: TOP

值

任何字符串

构成有效 IBM WebSphere MQ 主题字符串的字符串。将 IBM WebSphere MQ 用作与 WebSphere Application Server 配合使用的消息传递提供程序时，请指定与 WebSphere Application Server 中用于管理目的的主题已知名称相匹配的值。

相关概念

[主题字符串](#)

TRANSPORT

与队列管理器或代理的连接的性质。

适用对象

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :TRANSPORT

JMS 管理工具短名称 :TRAN

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setTransport 类型 ()
- MQConnectionFactory.getTransport 类型 ()

值

BIND

用于以绑定方式连接到队列管理器。这是管理工具的缺省值。

对于程序，请使用 WMQConstants.WMQ_CM_BINDINGS。

CLIENT

用于以客户机方式连接到队列管理器。

对于程序，请使用 WMQConstants.WMQ_CM_CLIENT。

直接

用于与不使用 HTTP 隧道的代理程序的实时连接。

对于程序，请使用 WMQConstants.WMQ_CM_DIRECT_TCPIP。

直接 (DIRECTHTTP)

用于使用 HTTP 隧道连接到代理程序的实时连接。仅支持 HTTP 1.0。

对于程序，请使用 WMQConstants.WMQ_CM_DIRECT_HTTP。

WILDCARDFORMAT

此属性确定要使用的通配符语法版本。

适用对象

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理工具长名称 :WILDCARDFORMAT

JMS 管理工具短名称 :WCFMT

以编程方式访问

Setter 方法/getter 方法

- MQConnectionFactory.setWildcardFormat()
- MQConnectionFactory.getWildcardFormat()

值

仅 TOPIC_ONLY

仅识别在代理版本 2 中使用的主题级别通配符。这是管理工具的缺省值。

对于程序, 请使用 WMQConstants.WMQ_WILDCARD_TOPIC_ONLY。

仅字符

仅识别在代理版本 1 中使用的字符通配符。

对于程序, 请使用 WMQConstants.WMQ_WILDCARD_CHAR_ONLY。

WebSphere MQ JMS 对象类的属性之间的依赖关系

某些属性的有效性取决于其他属性的特定值。

此依赖关系可能发生在以下属性组中:

- 客户机属性
- 与代理程序的实时连接的属性
- 退出初始化字符串

客户机属性

对于与队列管理器的连接, 仅当 TRANSPORT 具有值 CLIENT 时, 以下属性才相关:

- HOSTNAME
- 端口
- 通道
- LOCALADDRESS
- CCDTURL
- CCSID
- COMPHDR
- COMPMMSG
- RESEXIT
- RESEXITINIT
- SESEXIT
- SESEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE

- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

如果 TRANSPORT 具有值 BIND，那么不能使用管理工具来设置这些属性的值。

如果 TRANSPORT 的值为 CLIENT，那么 BROKERVER 属性的缺省值为 V1，而 PORT 属性的缺省值为 1414。如果显式设置 BROKERVER 或 PORT 的值，那么稍后对 TRANSPORT 值的更改不会覆盖您的选择。

与代理程序的实时连接的属性

仅当 TRANSPORT 的值为 DIRECT 或 DIRECTHTTP 时，以下属性才相关：

- BROKERVER
- CLIENTID
- DESCRIPTION
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- 多点广播 (仅支持 DIRECT)
- 端口
- PROXYHOSTNAME (仅支持 DIRECT)
- PROXYPORT (仅支持 DIRECT)

如果 TRANSPORT 的值为 DIRECT 或 DIRECTHTTP，那么 BROKERVER 属性的缺省值为 V2，并且 PORT 属性的缺省值为 1506。如果显式设置 BROKERVER 或 PORT 的值，那么稍后对 TRANSPORT 值的更改不会覆盖您的选择。

退出初始化字符串

请勿在不提供相应出口名称的情况下设置任何出口初始化字符串。出口初始化属性为：

- REEXITINIT
- SEEXITINIT
- SENDEXITINIT

例如，指定 REEXITINIT(myString) 而不指定 REEXIT(some.exit.classname) 会导致错误。

ENCODING 属性

ENCODING 属性包含三个子属性，采用 12 种可能的组合。

ENCODING 属性可以采用的有效值是从三个子属性构造的：

整型编码

正常或反转

十进制编码

正常或反转

浮点编码

IEEE 正常，IEEE 逆向或 z/OS

ENCODING 属性表示为具有以下语法的三字符字符串：

```
{N|R}{N|R}{N|R|3}
```

在此字符串中:

- N 表示正常
- R 表示反转
- 3 表示 z/OS
- 第一个字符表示 整数编码
- 第二个字符表示 十进制编码
- 第三个字符表示 浮点编码

这将为 ENCODING 属性提供包含 12 个可能值的集合。

还有一个附加值, 即字符串 NATIVE, 用于为 Java 平台设置相应的编码值。

以下示例显示了 ENCODING 的有效组合:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

JMS 对象的 SSL 属性

使用 SSLCIPHERSUITE 属性启用安全套接字层 (SSL) 加密。然后, 您可以使用其他几个属性来更改 SSL 加密的特征。

当您指定了传输 (CLIENT) 时, 可以使用 SSLCIPHERSUITE 属性来启用安全套接字层 (SSL) 加密通信。将此属性设置为 JSSE 提供程序提供的有效 CipherSuite ; 它必须与 CHANNEL 属性指定的 SVRCONN 通道上的 CipherSpec 匹配。

但是, CipherSpecs (在 SVRCONN 通道上指定) 和 CipherSuites (在 ConnectionFactory 对象上指定) 使用不同的命名方案来表示相同的 SSL 加密算法。如果在 SSLCIPHERSUITE 属性上指定了可识别的 CipherSpec 名称, 那么 JMSAdmin 将发出警告并将 CipherSpec 映射到其等效的 CipherSuite。请参阅 [JMS 中的 SSL CipherSpecs 和 CipherSuites](#), 以获取 WebSphere MQ 和 JMSAdmin 识别的 CipherSpecs 列表。

如果需要连接以使用 IBM Java JSSE FIPS 提供程序 (IBMJSSEFIPS) 支持的 CipherSuite, 请将连接工厂的 SSLFIPSREQUIRED 属性设置为 YES。此属性的缺省值为 NO, 这意味着连接可以使用任何受支持的 CipherSuite。如果未设置 SSLCIPHERSUITE, 那么将忽略该属性。

SSLPEERNAME 与 SSLPEER 参数的格式匹配, 可以在通道定义上设置此格式。它是由逗号或分号分隔的属性名称/值对的列表。例如:

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSphere)
```

名称和值集构成 专有名称。有关专有名称及其用于 WebSphere MQ 的更多详细信息, 请参阅 [安全性](#)。

给定的示例检查服务器在连接时提供的标识证书。要使连接成功, 证书必须具有以 QMGR 开头的公共名称。并且必须至少具有两个组织单元名称, 其中第一个是 IBM 和第二个 WEBSphere。检查不区分大小写。

如果未设置 SSLPEERNAME, 那么不会执行此类检查。如果未设置 SSLCIPHERSUITE, 那么将忽略 SSLPEERNAME。

SSLCRL 属性指定零个或多个 CRL (证书撤销列表) 服务器。使用此属性需要 Java 2 v1.4 上的 JVM。这是格式为以下的条目的空格分隔列表:

```
ldap://hostname:[port]
```

(可选) 后跟单个 /。如果省略 port, 那么将采用缺省 LDAP 端口 389。在连接时, 将针对指定的 CRL 服务器检查服务器提供的 SSL 证书。有关 CRL 安全性的更多信息, 请参阅 [安全性](#)。

如果未设置 SSLCRL, 那么不会执行此类检查。如果未设置 SSLCIPHERSUITE, 那么将忽略 SSLCRL。

SSLRESETCOUNT 属性表示在重新协商用于加密的密钥之前，连接发送和接收的总字节数。发送的字节数是加密之前的字节数，接收的字节数是解密之后的字节数。字节数还包括 WebSphere MQ JMS 类发送和接收的控制信息。

例如，要使用在流动的数据达到 4 MB 后重新协商的密钥配置 ConnectionFactory 对象（可用于创建通过启用 SSL 的 MQI 通道的连接），请向 JMSAdmin 发出以下命令：

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

如果 SSLRESETCOUNT 的值为零 (缺省值)，那么永远不会重新协商密钥。如果未设置 SSLCIPHERSUITE，那么将忽略 SSLRESETCOUNT 属性。

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或默示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以以书面形式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

知识产权许可
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 063-8506 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗示的）保证，包括但不限于暗示的有关非侵权，适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗示的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Corporation
软件互操作性协调员，部门 49XA
北纬 3605 号公路
罗切斯特，明尼苏达州 55901
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含日常商业运作所使用的数据和报表的示例。为了尽可能全面地说明这些数据和报表，这些示例包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有任何雷同，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或默示这些程序的可靠性、可维护性或功能。

如果您正在查看本信息的软拷贝，图片和彩色图例可能无法显示。

编程接口信息

编程接口信息 (如果提供) 旨在帮助您创建用于此程序的应用软件。

本书包含有关允许客户编写程序以获取 IBM WebSphere MQ 服务的预期编程接口的信息。

但是，该信息还可能包含诊断、修改和调优信息。提供诊断、修改和调优信息是为了帮助您调试您的应用程序软件。

要点: 请勿将此诊断，修改和调整信息用作编程接口，因为它可能会发生更改。

商标

IBM 徽标 ibm.com 是 IBM Corporation 在全球许多管辖区域的商标。当前的 IBM 商标列表可从 Web 上的“Copyright and trademark information”www.ibm.com/legal/copytrade.shtml 获取。其他产品和服务名称可能是 IBM 或其他公司的商标。

Microsoft 和 Windows 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 Open Group 在美国和其他国家或地区的注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

此产品包含由 Eclipse 项目 (<http://www.eclipse.org/>) 开发的软件。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其附属公司的商标或注册商标。



部件号:

(1P) P/N: